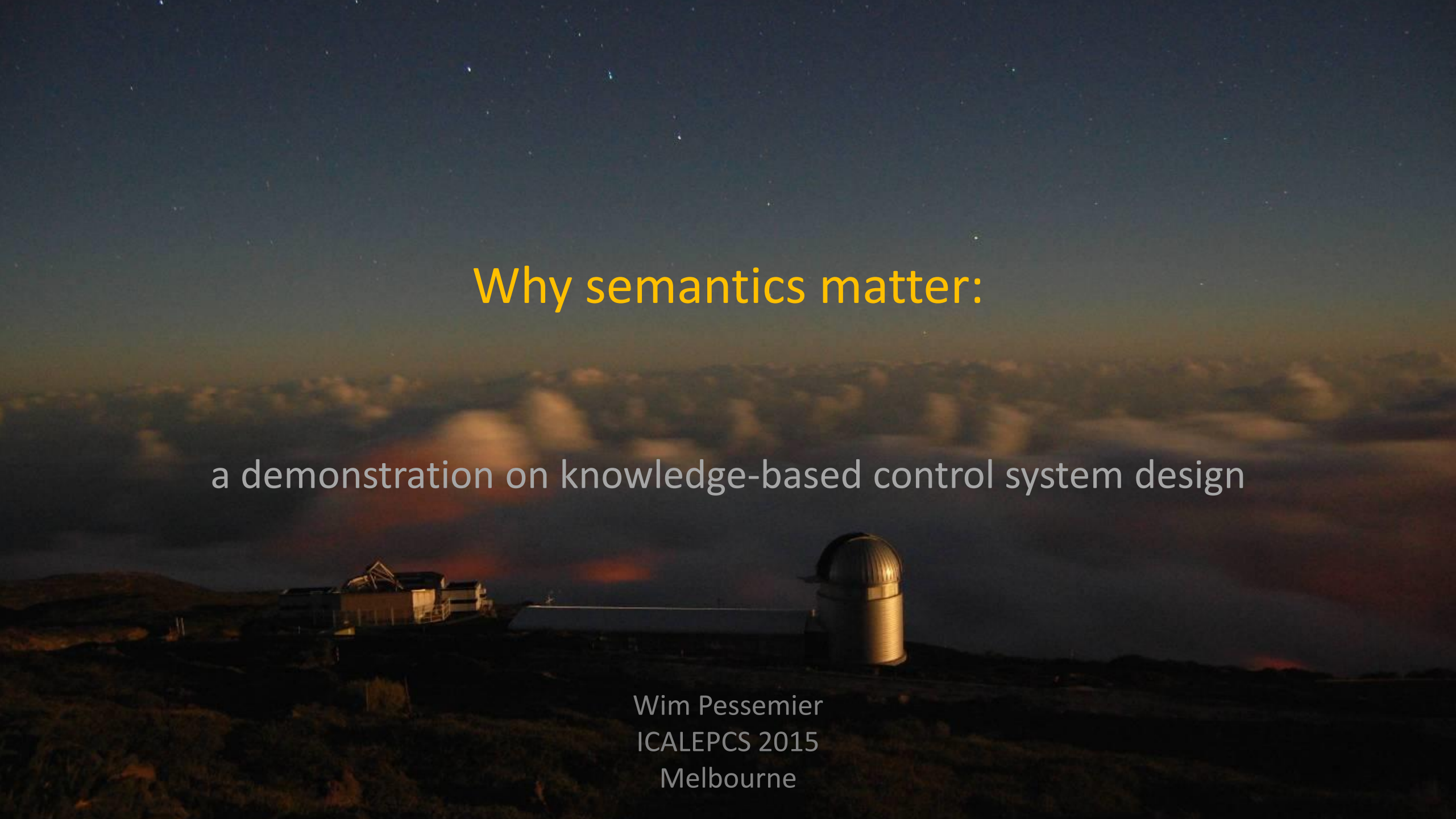
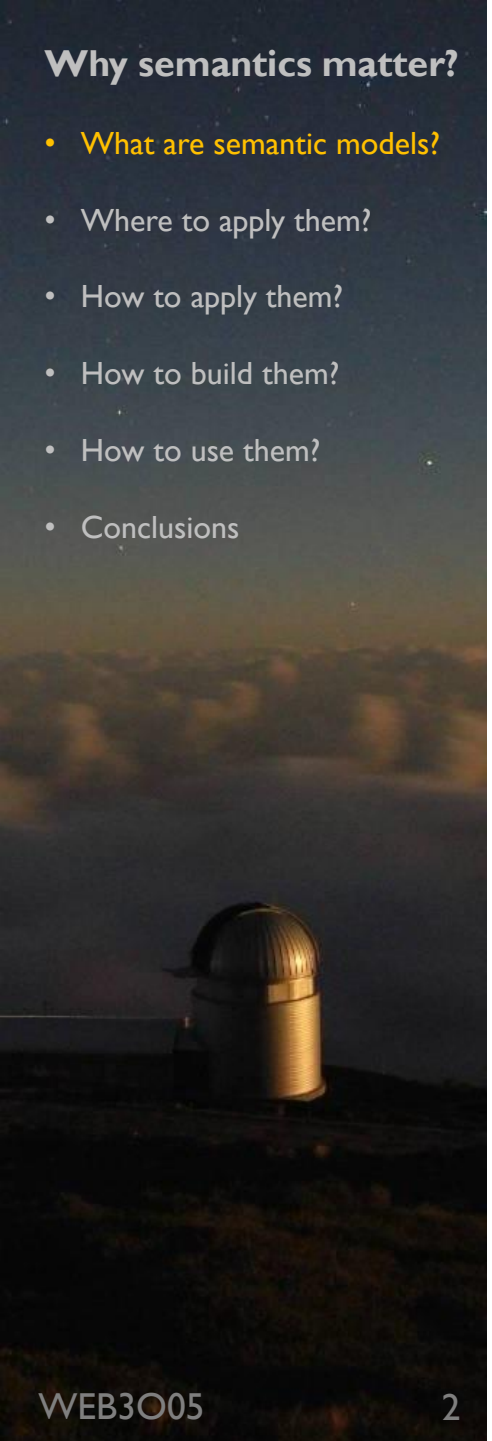


Why semantics matter:

a demonstration on knowledge-based control system design

Wim Pessemier
ICALEPCS 2015
Melbourne



What are semantic models?

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → meaning (semantics)

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → **meaning (semantics)**

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → **meaning (semantics)**

GENERIC

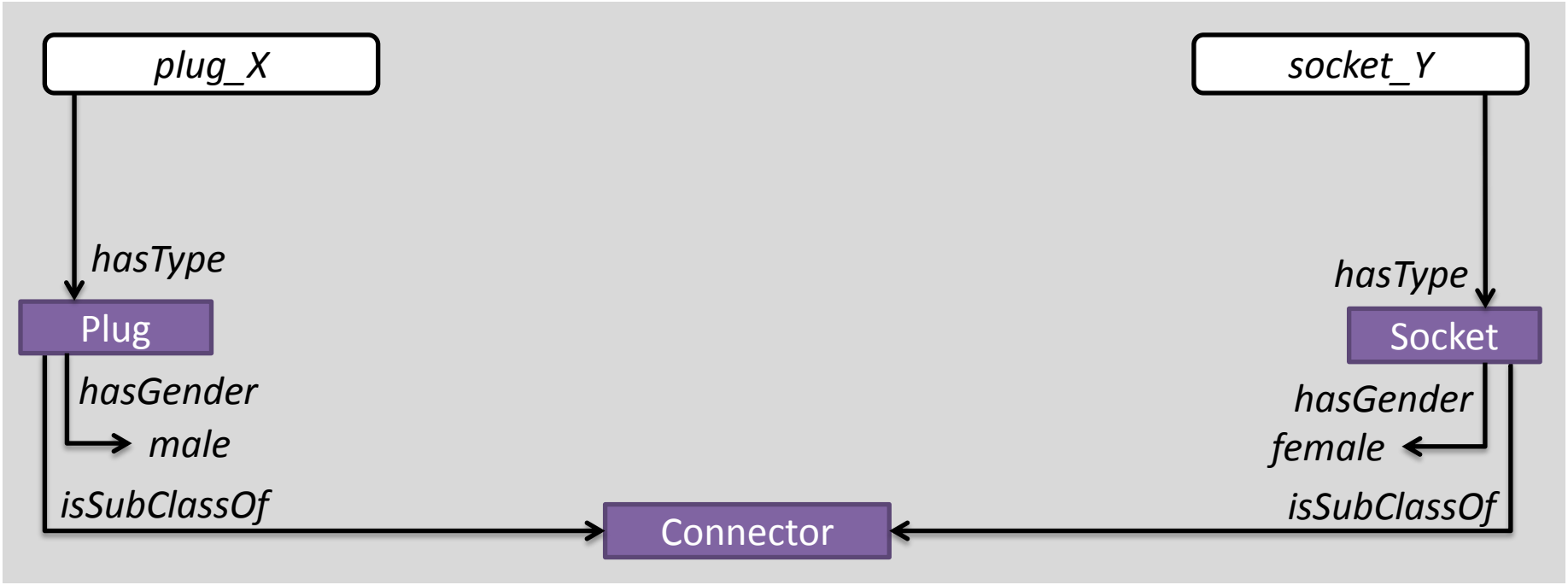
hasType

ELECTRIC

Plug
Socket

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → **meaning (semantics)**



GENERIC

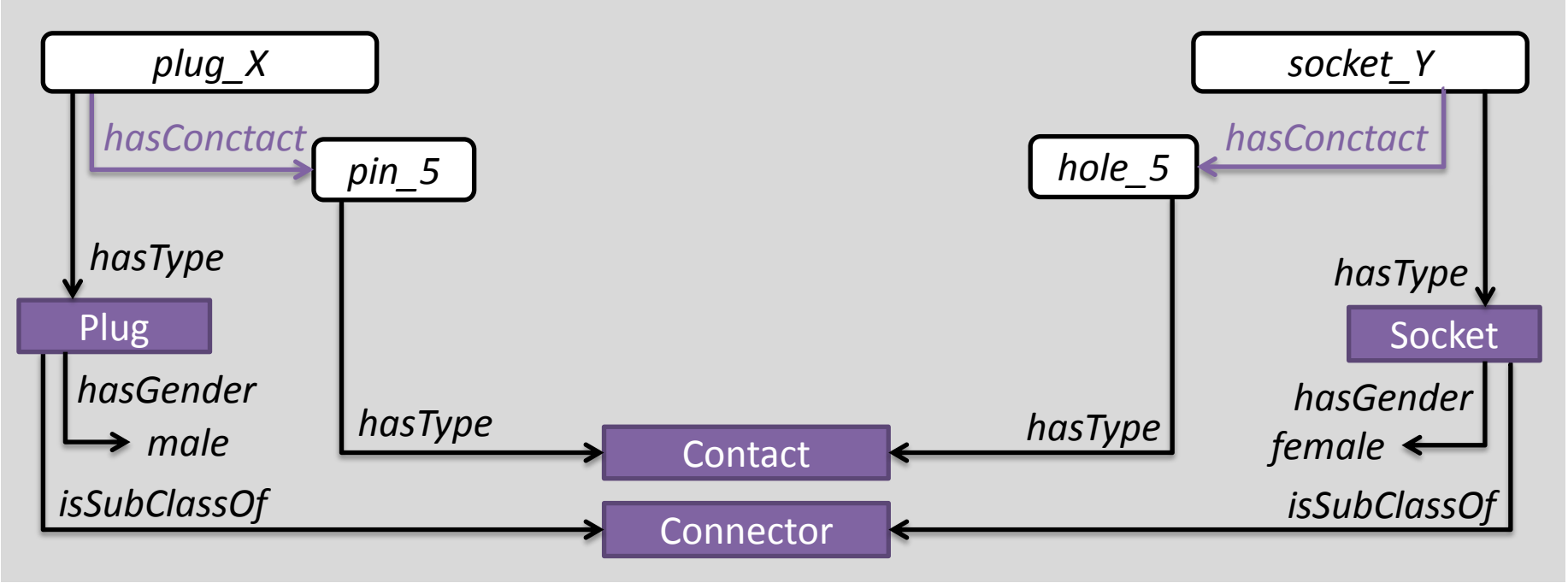
hasType

ELECTRIC

Plug
Socket

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → **meaning (semantics)**



GENERIC

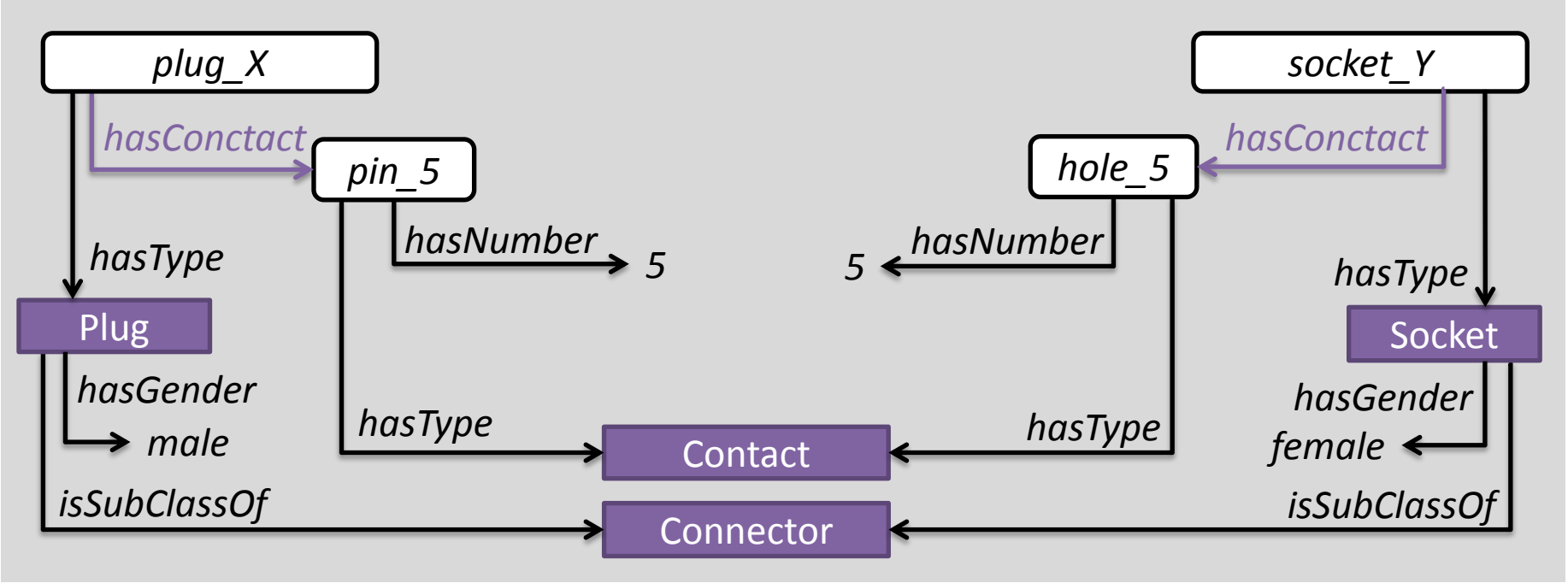
hasType *isSubClassOf*
hasGender

ELECTRIC

Plug Connector
Socket *hasContact*

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → **meaning (semantics)**



GENERIC

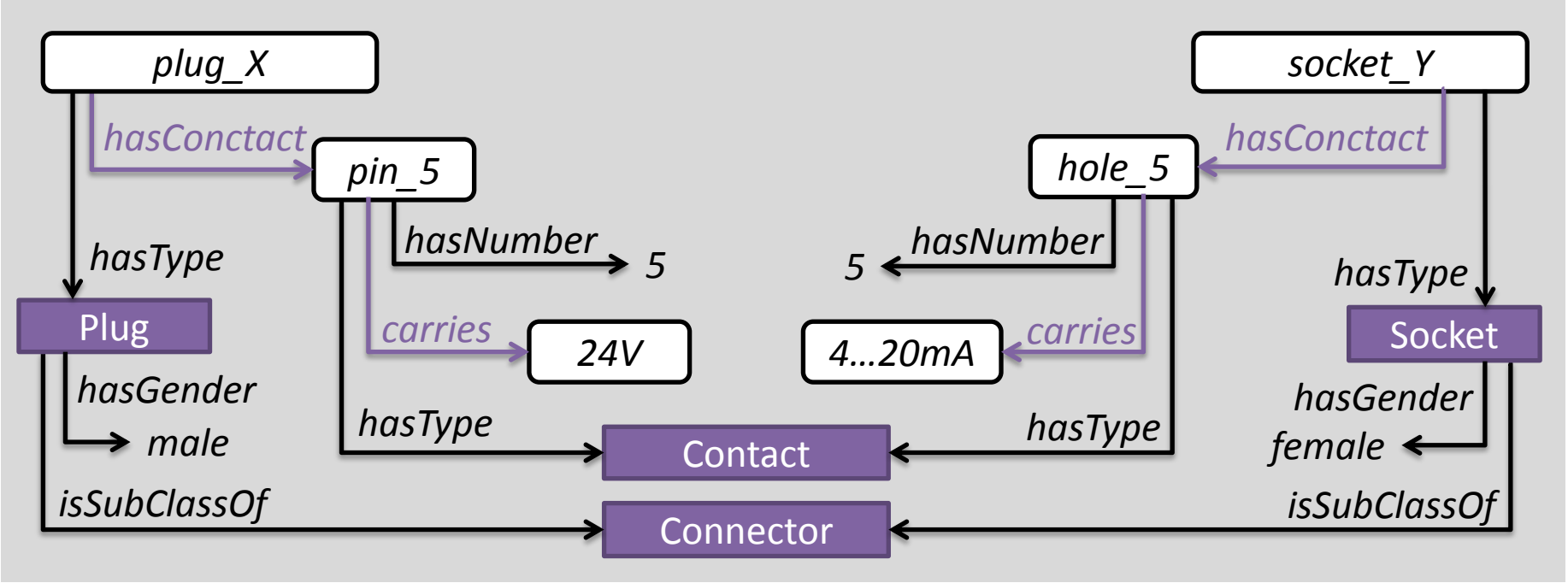
hasType *isSubClassOf*
hasGender *hasNumber*

ELECTRIC

Plug Connector Contact
Socket *hasContact*

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → **meaning (semantics)**



GENERIC

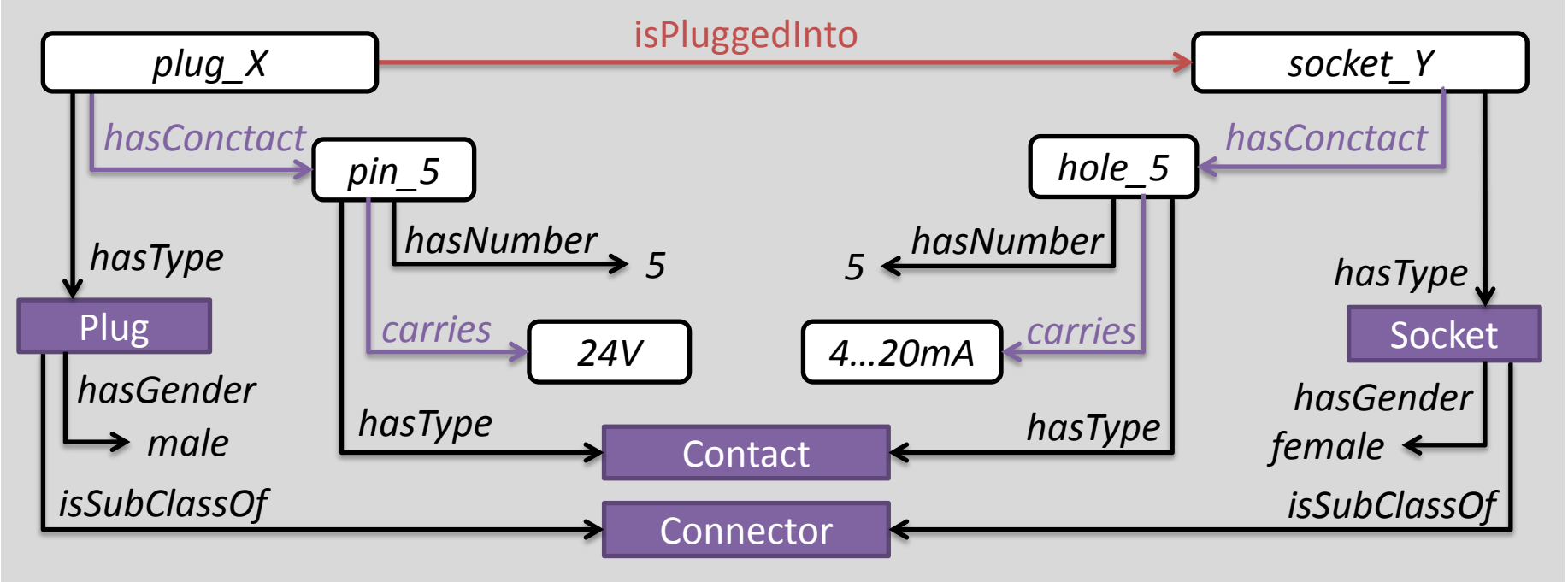
hasType *isSubClassOf*
hasGender *hasNumber*

ELECTRIC

Plug Connector Contact
Socket *hasContact* *carries*

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → **meaning (semantics)**



GENERIC

hasType *isSubClassOf*
hasGender *hasNumber*

ELECTRIC

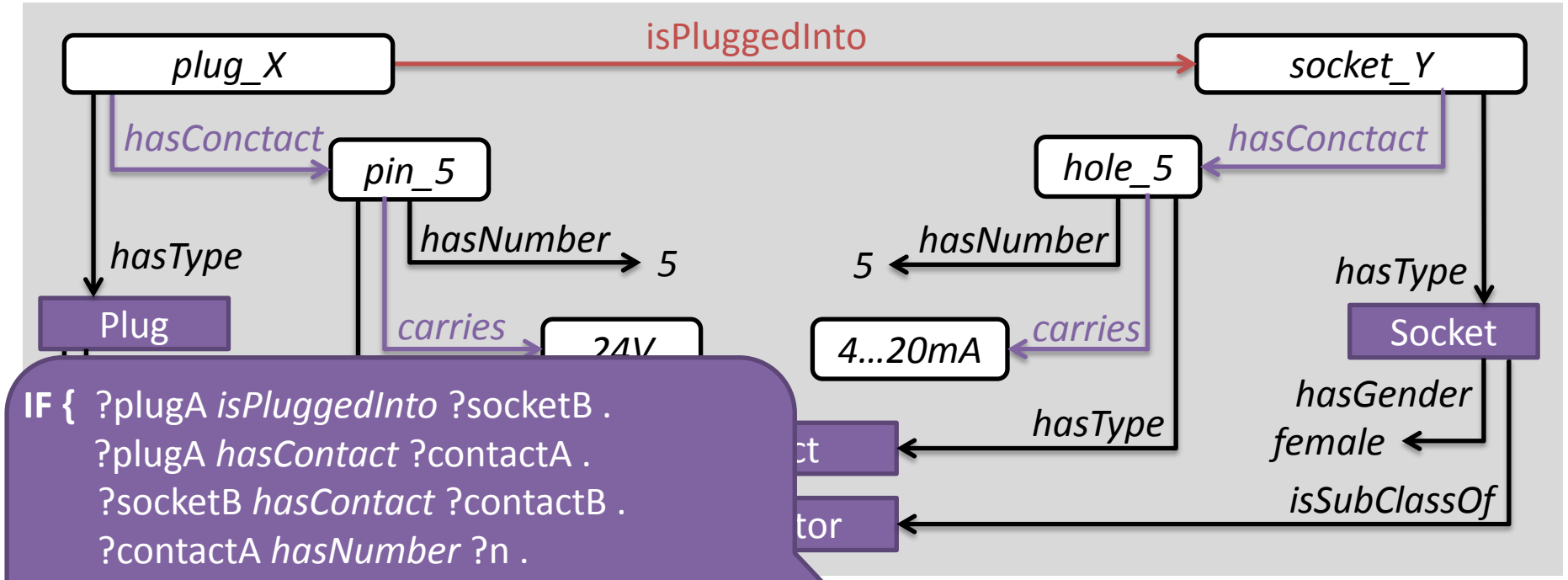
Plug Connector Contact
Socket *hasContact* *carries*

MECHANIC

isPluggedInto

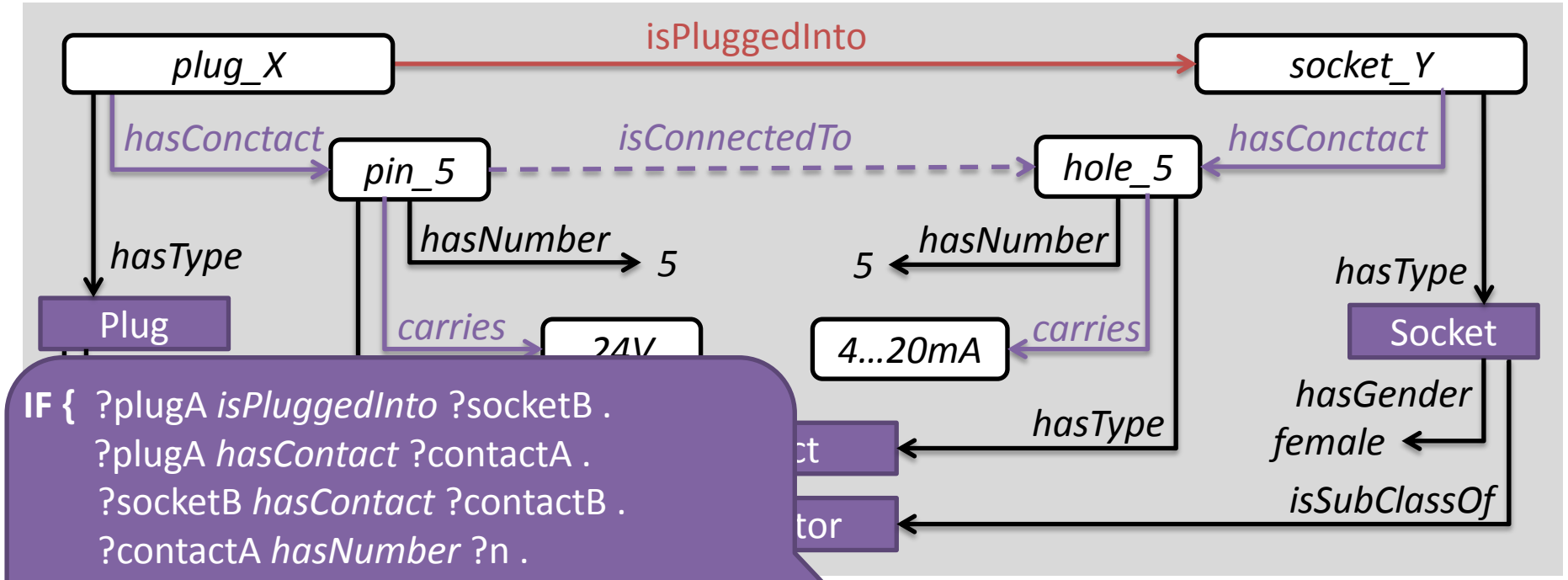
What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → **meaning (semantics)**



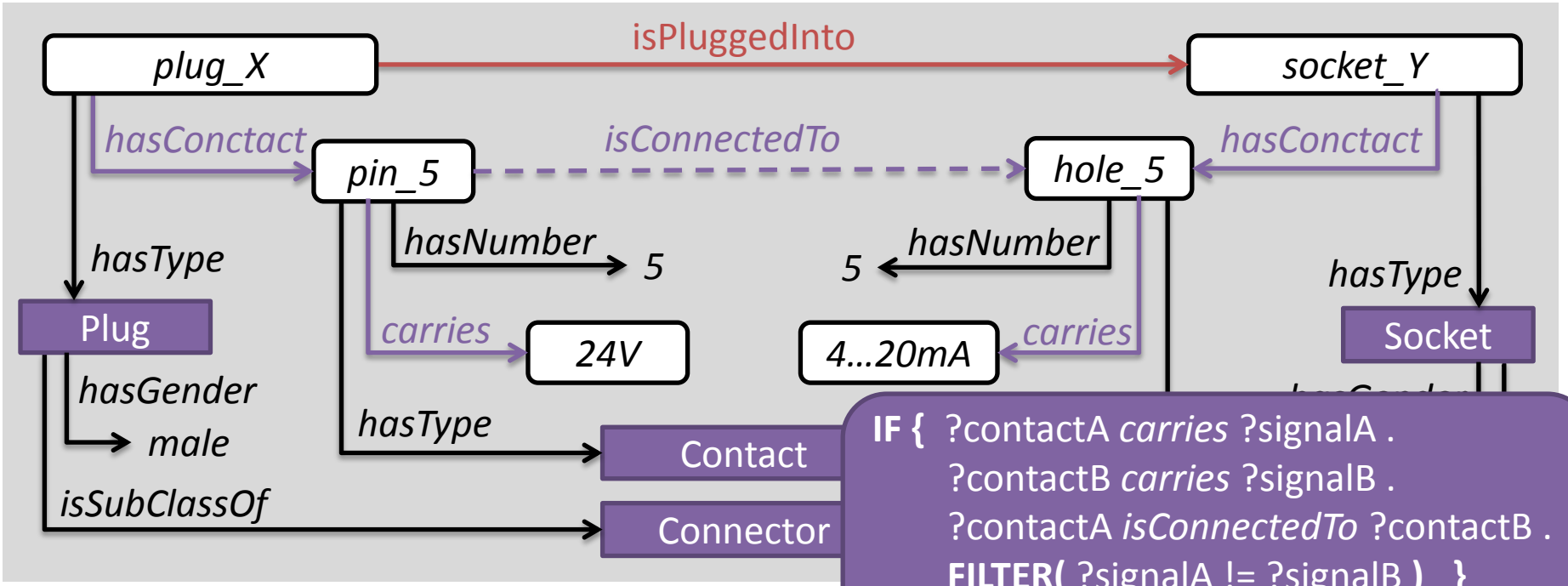
What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → **meaning (semantics)**



What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → **meaning (semantics)**



GENERIC

hasType *isSubClassOf*
hasGender *hasNumber*

ELECTRIC

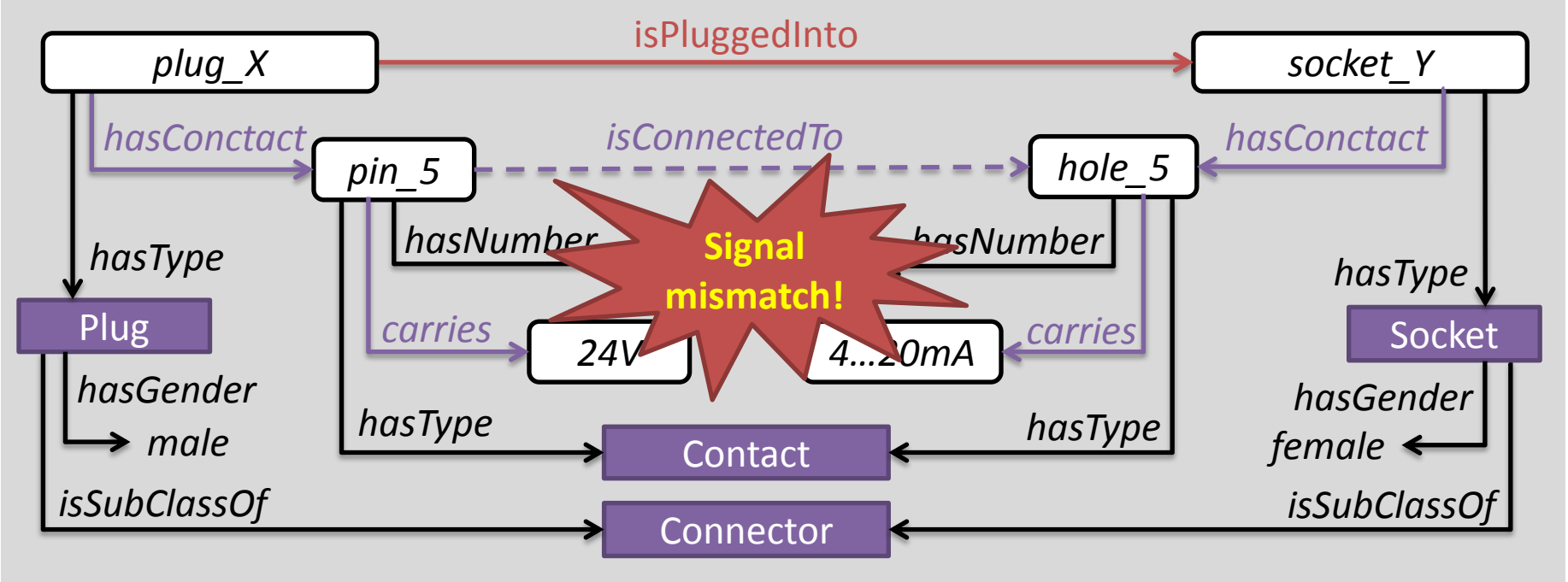
Plug Connector Contact
Socket *hasContact* *carries*

MECHANIC

isPluggedInto

What are semantic models?

- Models that describe
  - pieces of information (data, descriptions)
  - their relations → **meaning (semantics)**



GENERIC

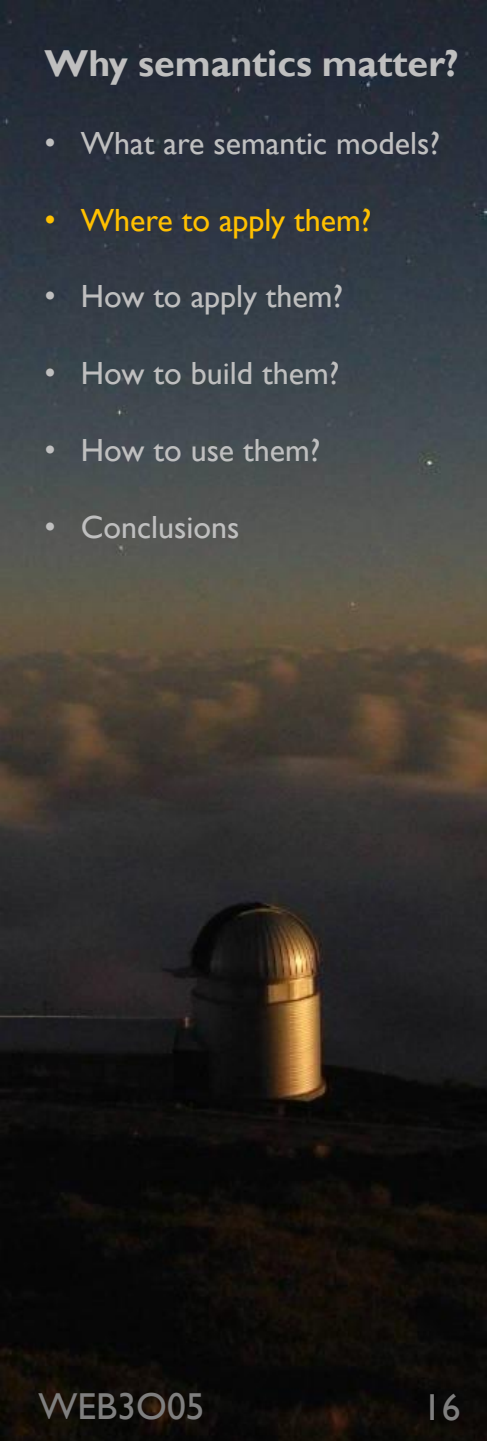
hasType *isSubClassOf*
hasGender *hasNumber*

ELECTRIC

Plug Connector Contact
Socket *hasContact* *carries*

MECHANIC

isPluggedInto



Where to apply them?

Where to apply them?

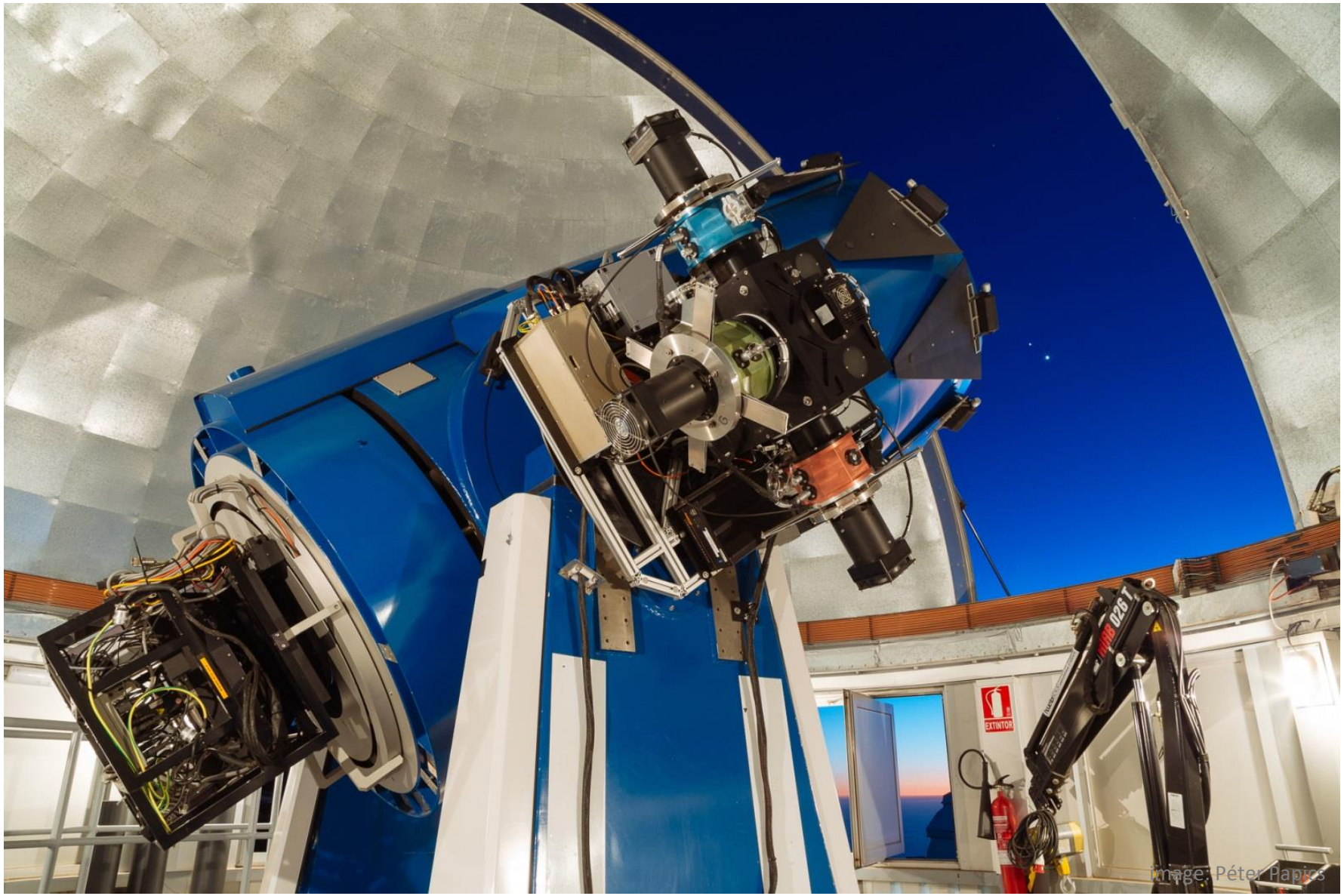
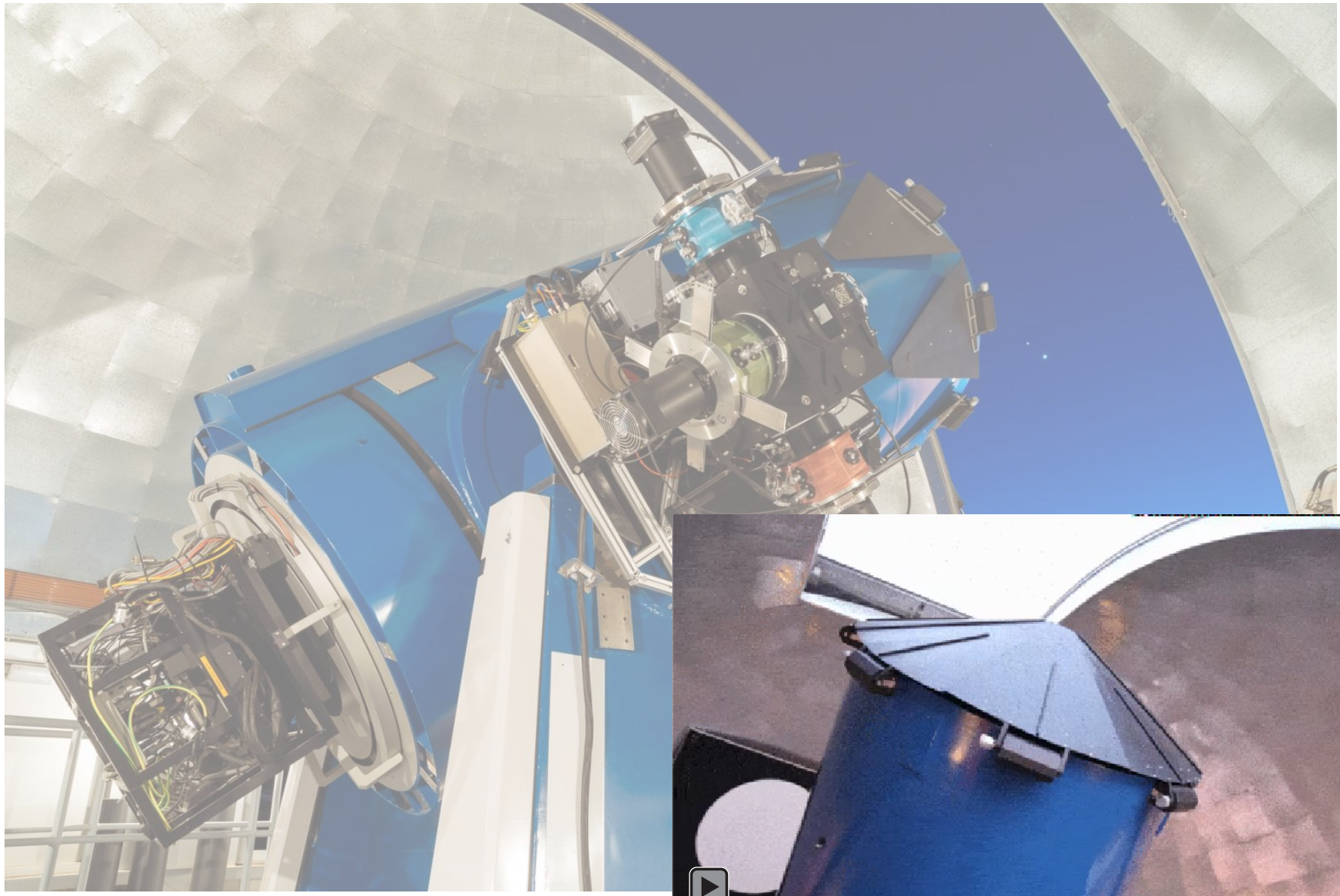
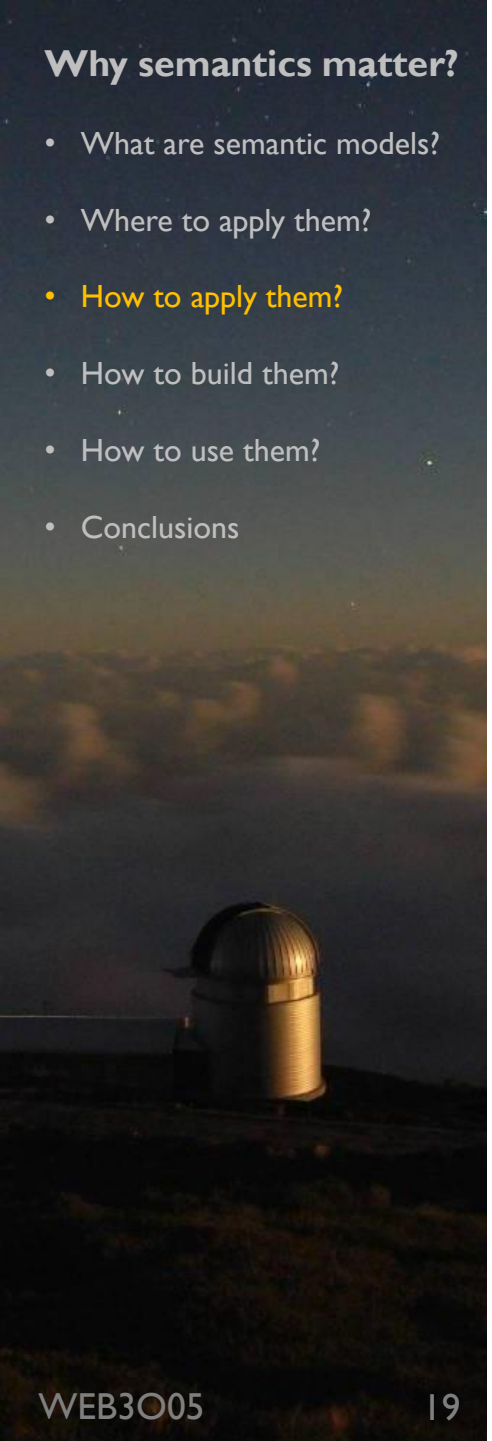


Image: Péter Papics

Where to apply them?

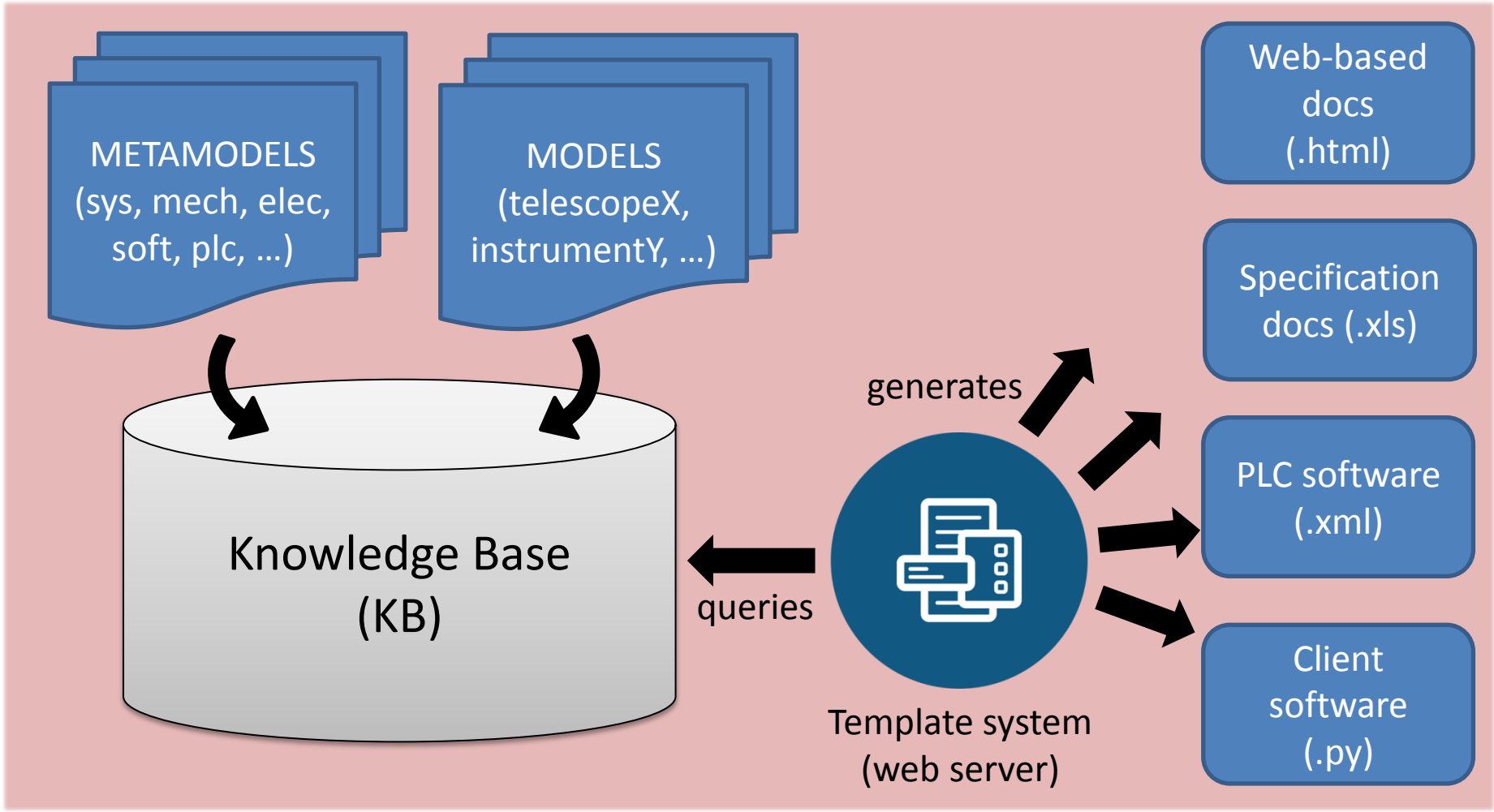




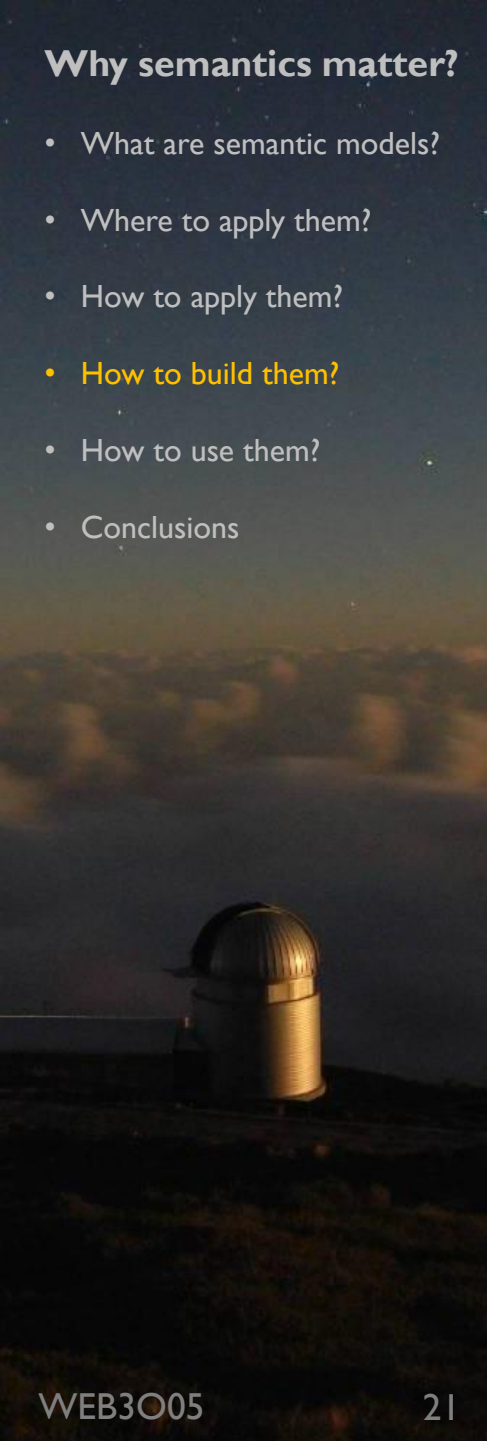
How to apply them?

How to apply them?

- Put them in a Knowledge Base and extract information!



OntoManager

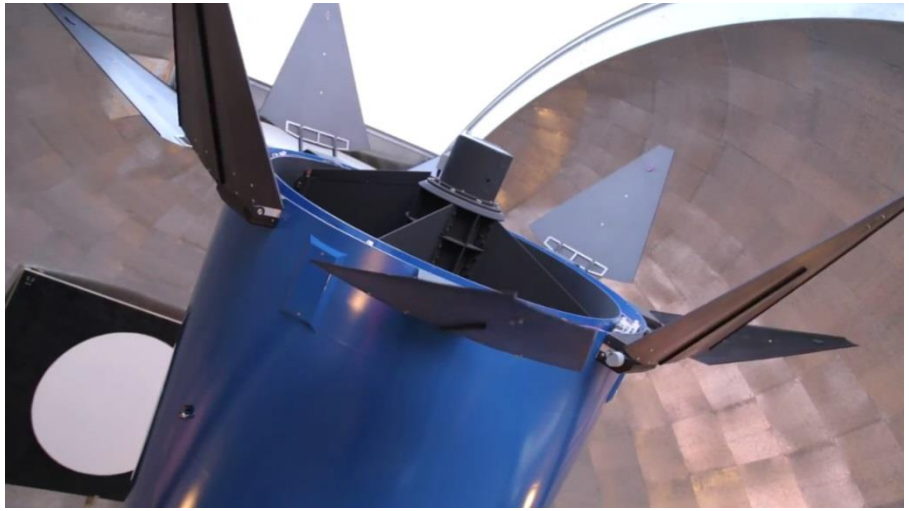


How to build them?

How to build them?

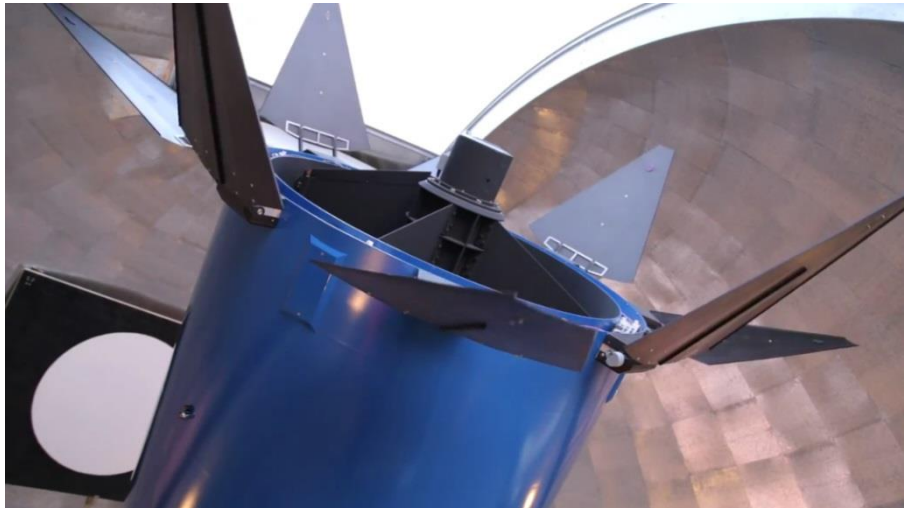
How to build them?

- Using an existing modeling language?
  - UML, SysML, … : semantics not sufficiently formal
  - Modeling languages have no "programming" capabilities (loops, functions, if-then, …)



How to build them?

- Using an existing modeling language?
  - UML, SysML, … : semantics not sufficiently formal
  - Modeling languages have no "programming" capabilities (loops, functions, if-then, …)

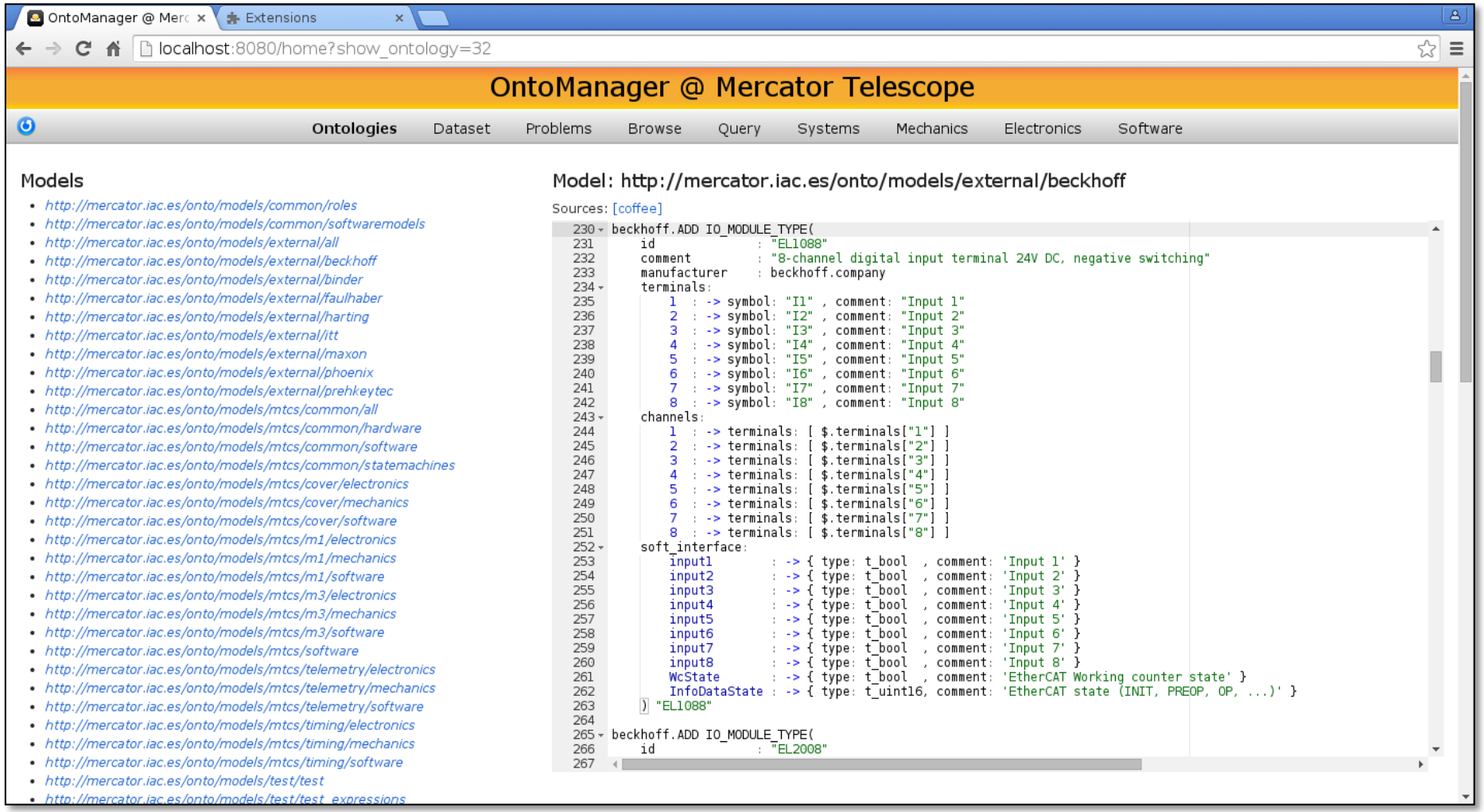


- Using a Domain Specific Language (DSL)?
  - Internal DSL called Ontoscript
  - Based on coffeescript (~javascript)
  - Idea "adopted" from the Giant Magellan Telescope project [1]

[1] J. M. Filgueira, "GMT software and controls overview", Proc. SPIE 8451, Amsterdam, July 2012, 845111

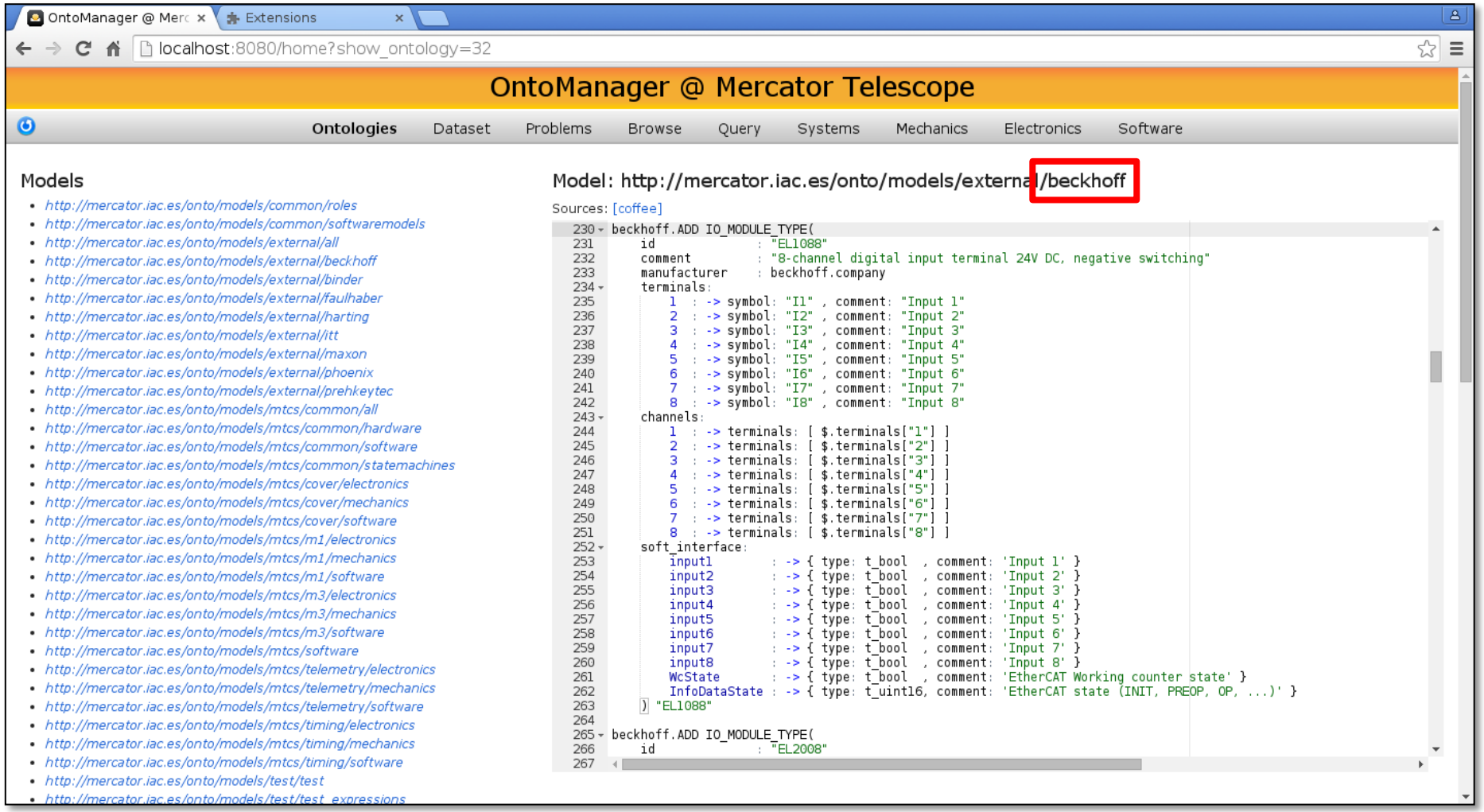
How to build them?

- Example: model of an I/O module **type**



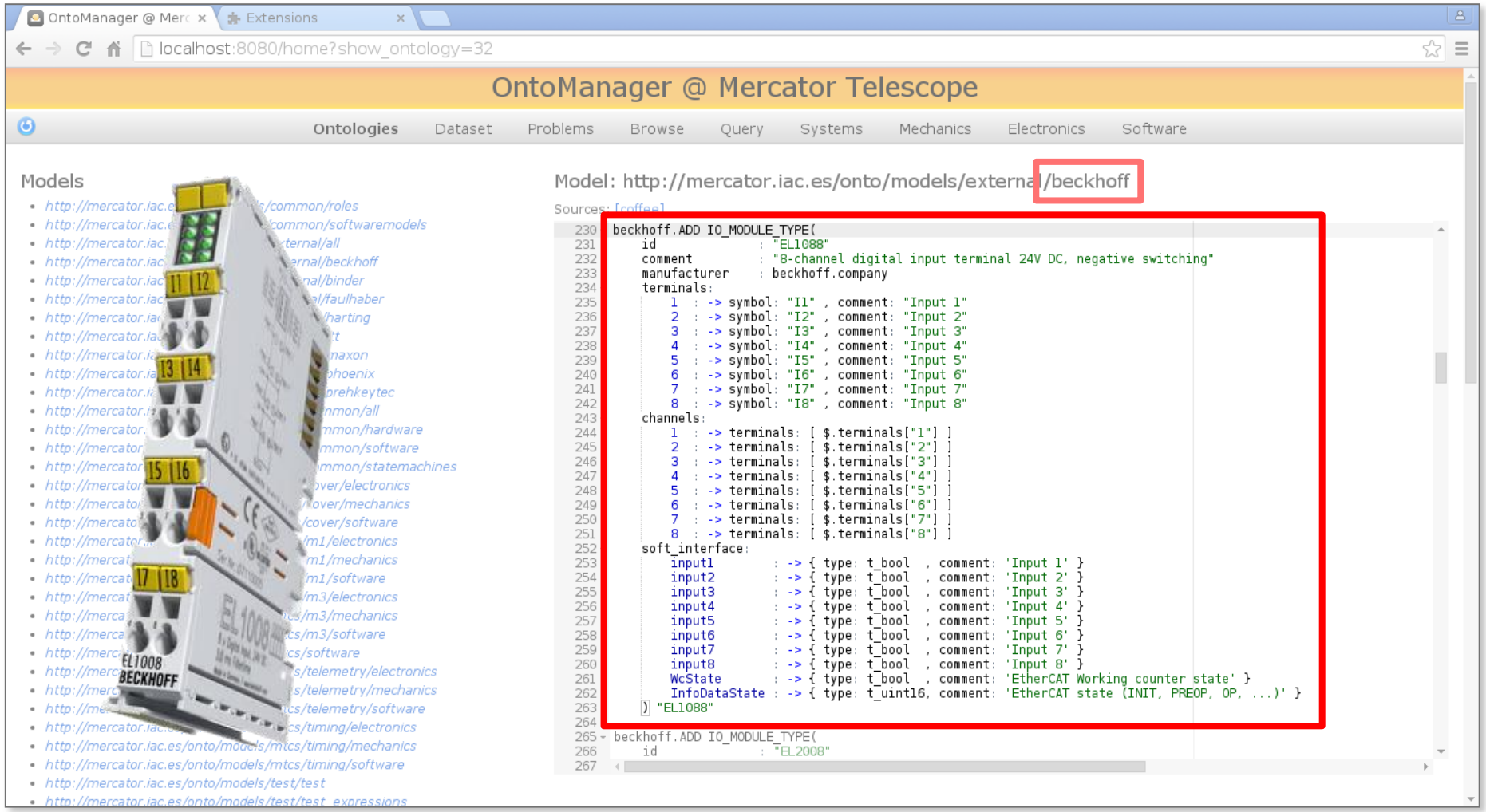
How to build them?

- Example: model of an I/O module **type**



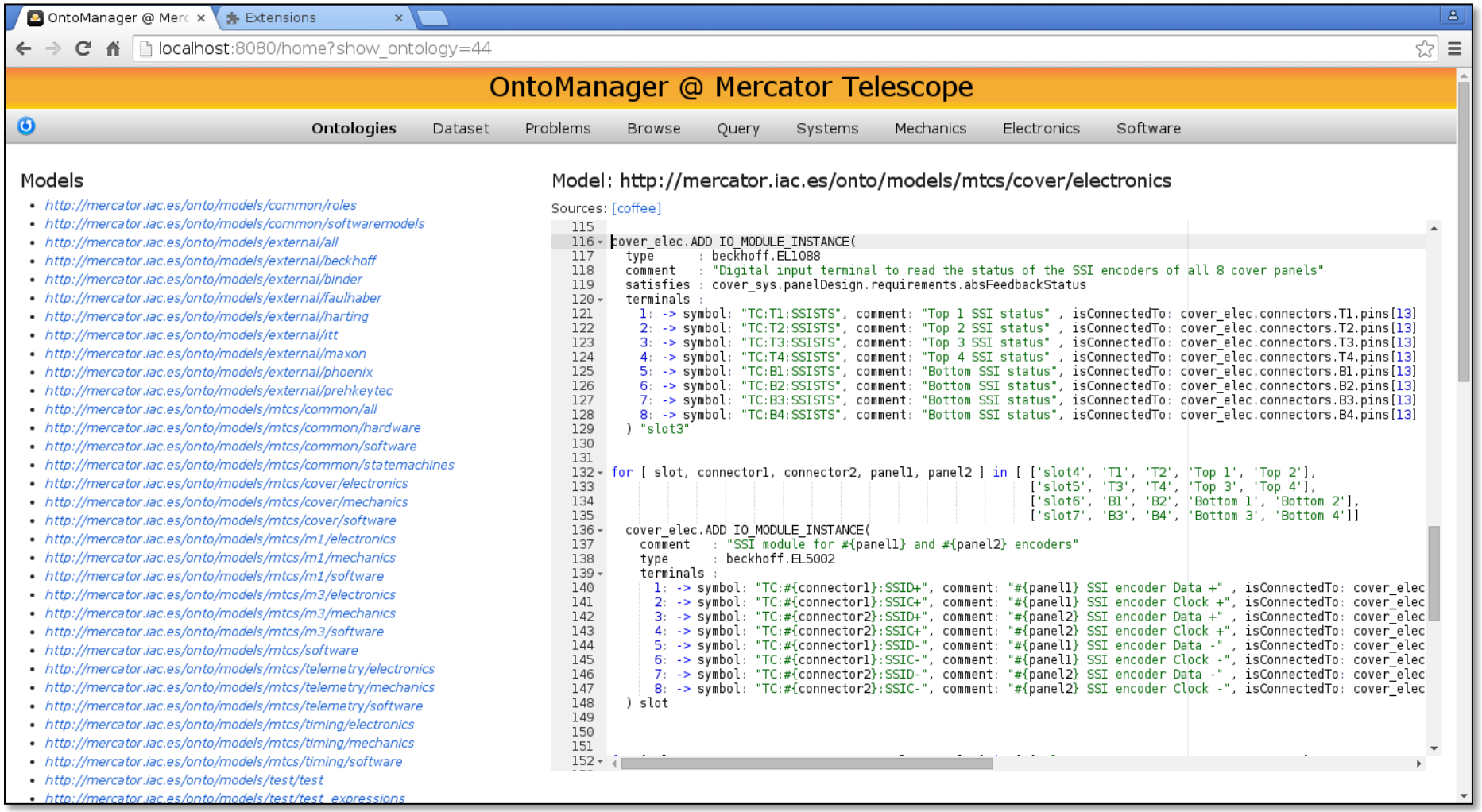
How to build them?

- Example: model of an I/O module **type**



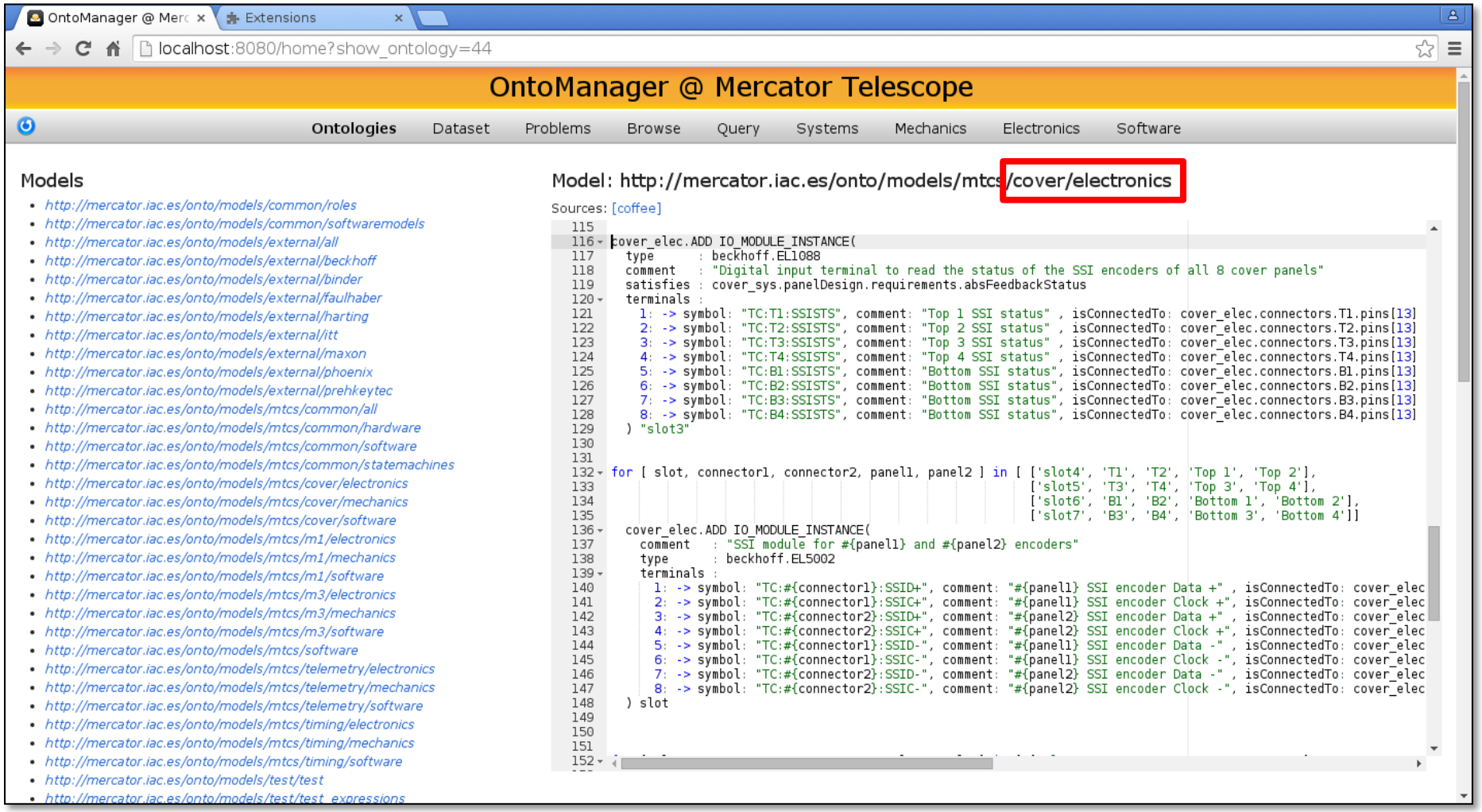
How to build them?

- Example: model of an I/O module **instance**



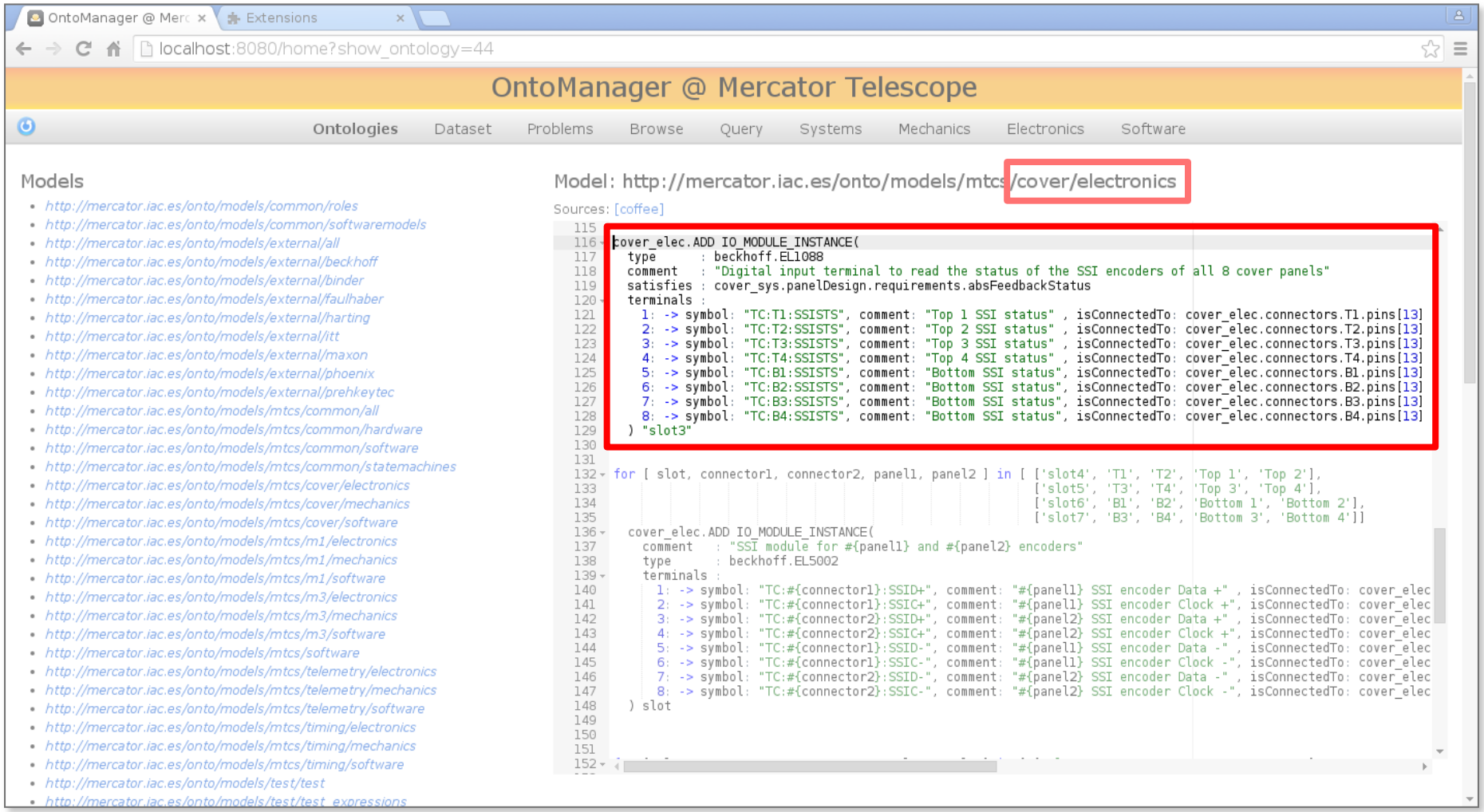
How to build them?

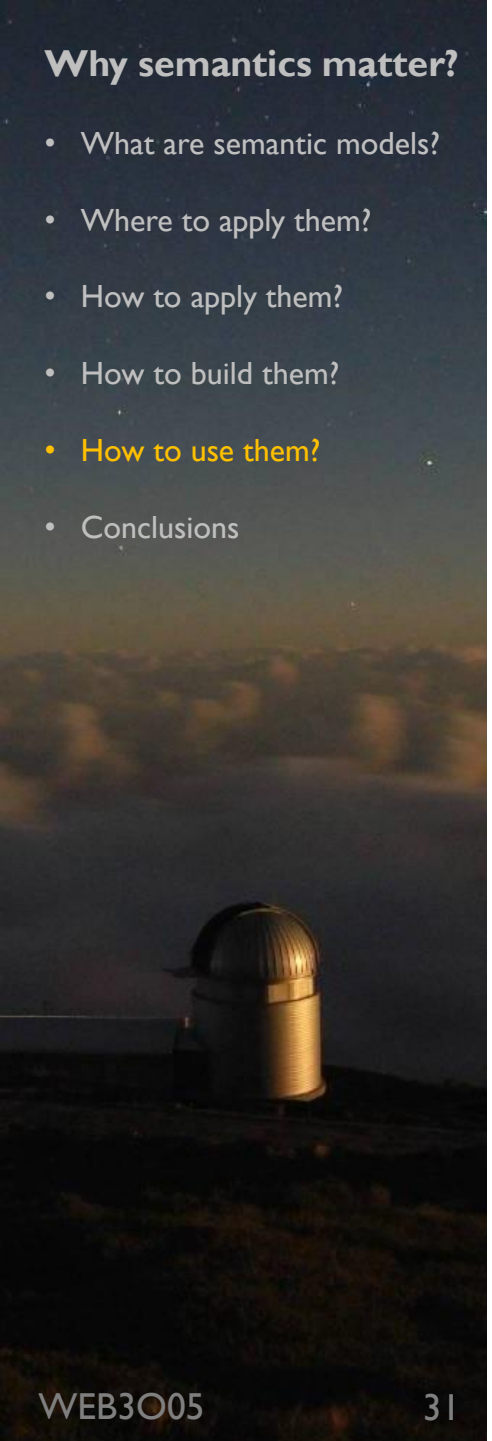
- Example: model of an I/O module **instance**



How to build them?

- Example: model of an I/O module **instance**





How to use them?

Electrical design

Electrical design

Electrical design

Electrical design

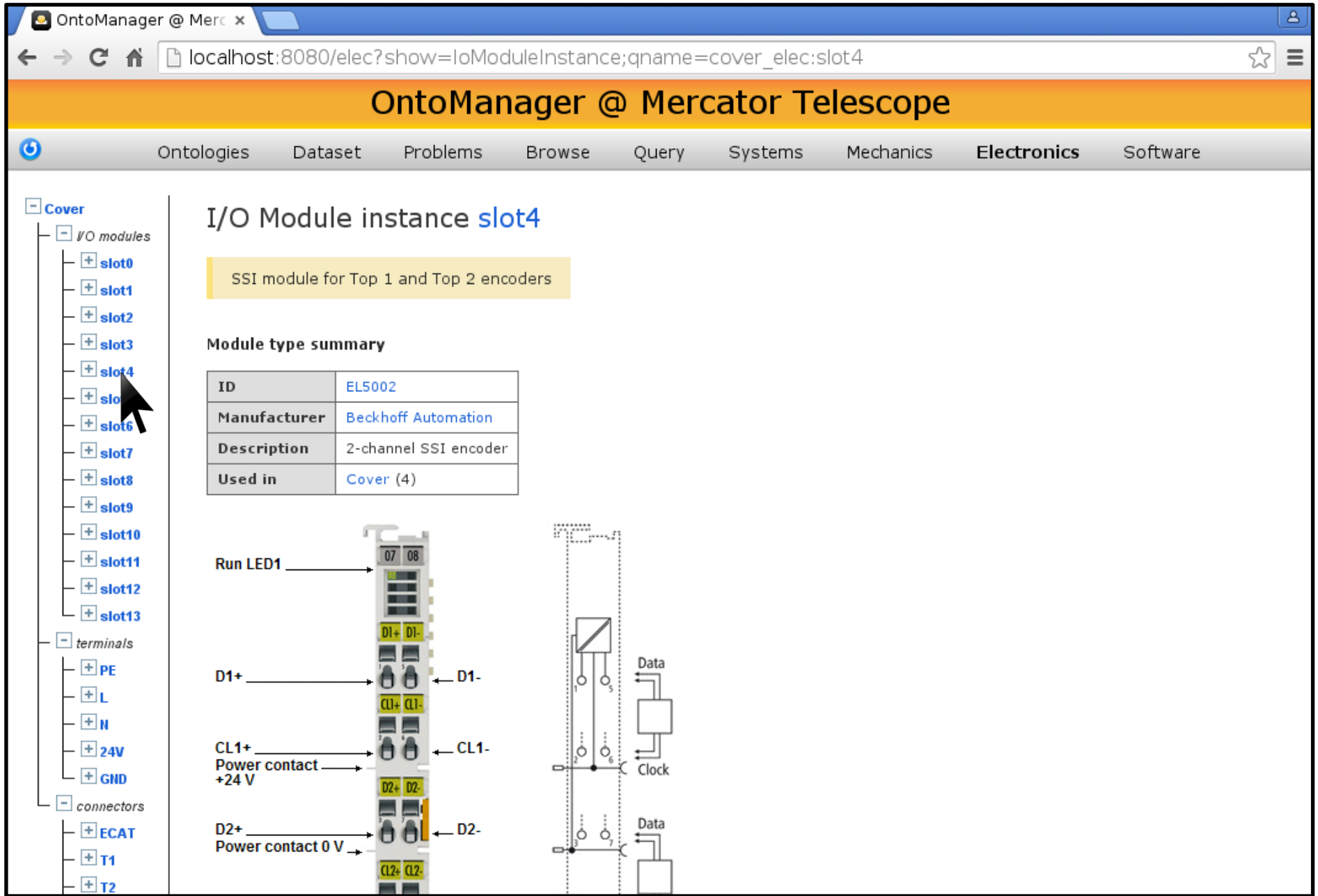
Electrical design

Electrical design

Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Electrical design

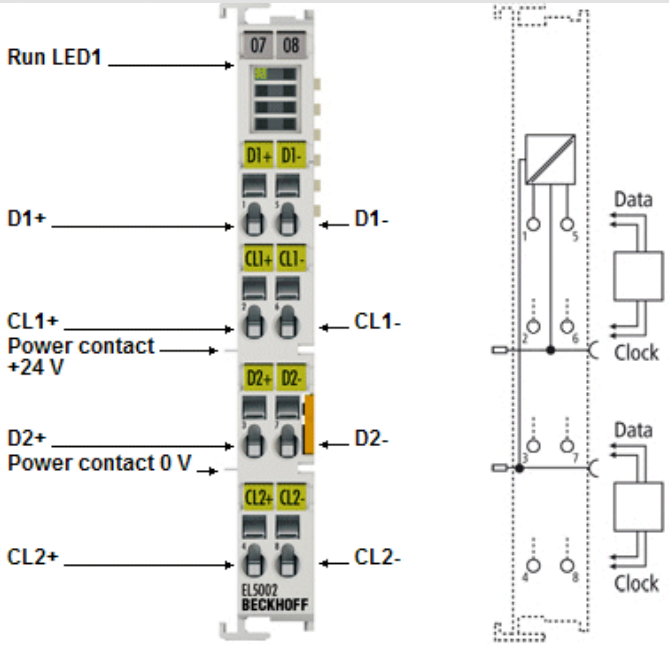


Electrical design

OntoManager @ Merc

localhost:8080/elec?show=IoModuleInstance;qname=cover_elec:slot4

- slot10
- slot11
- slot12
- slot13
- terminals
  - PE
  - L
  - N
  - 24V
  - GND
- connectors
  - ECAT
  - T1
  - T2
  - T3
  - T4
  - B1
  - B2
  - B3
  - B4
- M1
- M3
- Telemetry
- Timing



Connections

Type (EL5002)				Instance		
Channel	Terminal	Symbol	Description	Symbol	Description	Connected to
1	1	D1+	Channel 1: Data +	TC:T1:SSID+	Top 1 SSI encoder Data +	Connector T1 : pin 6
	2	CL1+	Channel 1: Clock +	TC:T1:SSIC+	Top 1 SSI encoder Clock +	Connector T1 : pin 7
	5	D1-	Channel 1: Data -	TC:T1:SSID-	Top 1 SSI encoder Data -	Connector T1 : pin 14
	6	CL1-	Channel 1: Clock -	TC:T1:SSIC-	Top 1 SSI encoder Clock -	Connector T1 : pin 15
2	3	D2+	Channel 2: Data +	TC:T2:SSID+	Top 2 SSI encoder Data +	Connector T2 : pin 6
	4	CL2+	Channel 2: Clock +	TC:T2:SSIC+	Top 2 SSI encoder Clock +	Connector T2 : pin 7
	7	D2-	Channel 2: Data -	TC:T2:SSID-	Top 2 SSI encoder Data -	Connector T2 : pin 14
	8	CL2-	Channel 2: Clock -	TC:T2:SSIC-	Top 2 SSI encoder Clock -	Connector T2 : pin 15

Electrical design

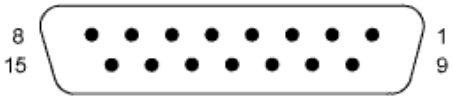
Electrical design

OntoManager @ Merc ×

localhost:8080/elec?show=ConnectorInstance;qname=cover_elec:connectors.T1

- slot5
- slot6
- slot7
- slot8
- slot9
- slot10
- slot11
- slot12
- slot13
- terminals
  - PE
  - L
  - N
  - 24V
  - GND
- connectors
  - ECAT
  - T1
  - T2
  - T3
  - T4
  - B1
  - B2
  - B3
  - B4
- M1
- M3
- Telemetry
- Timing

Description	D-sub 15 female connector
Fits to	D-sub 15 M
Used in	Cover (8), M1 (1), M3 (2)



DA-15S (Female Socket Front View)

Connections

Type (D-sub 15 F)			Instance		
Pin	**Symbol**	**Description**	**Symbol**	**Description**	**Connected to**
1	1	Pin 1	TC:T1:GND HM	Top 1 GND of holding magnet	Cover : terminal GND
2	2	Pin 2	TC:T1:GND MOT	Top 1 GND of motor	Cover : terminal GND
3	3	Pin 3	TC:T1:MMON	Top 1 motor monitor	
4	4	Pin 4	TC:T1:MDIR	Top 1 motor direction	I/O module slot1 : terminal 1
5	5	Pin 5	TC:T1:GND ENC	Top 1 GND of encoder	Cover : terminal GND
6	6	Pin 6	TC:T1:SSID+	Top 1 SSI Data +	I/O module slot4 : terminal 1
7	7	Pin 7	TC:T1:SSIC+	Top 1 SSI Clock +	I/O module slot4 : terminal 2
8	8	Pin 8	TC:T1:PE	Top 1 Earth	Cover : terminal PE
9	9	Pin 9	TC:T1:+24V HM	Top 1 +24V of holding magnet	I/O module slot12 : terminal 2
10	10	Pin 10	TC:T1:+24V MOT	Top 1 +24V of motor	I/O module slot8 : terminal 2
11	11	Pin 11	TC:T1:MSPEED	Top 1 motor speed	I/O module slot2 : terminal 1
12	12	Pin 12	TC:T1:+24V ENC	Top 1 +24V of encoder	Cover : terminal 24V
13	13	Pin 13	TC:T1:SSISTS	Top 1 SSI status	I/O module slot3 : terminal 1
14	14	Pin 14	TC:T1:SSID-	Top 1 SSI Data -	I/O module slo terminal 5
15	15	Pin 15	TC:T1:SSIC-	Top 1 SSI Clock -	I/O module slot4 : terminal 6

Electrical design

OntoManager @ Merc ×

localhost:8080/elec?show=IoModuleInstance;qname=cover_elec:slot3

OntoManager @ Mercator Telescope

Ontologies | Dataset | Problems | Browse | Query | Systems | Mechanics | **Electronics** | Software

- Cover
  - *I/O modules*
    - slot0
    - slot1
    - slot2
    - slot3
    - slot4
    - slot5
    - slot6
    - slot7
    - slot8
    - slot9
    - slot10
    - slot11
    - slot12
    - slot13
  - *terminals*
    - PE
    - L
    - N
    - 24V
    - GND
  - *connectors*
    - ECAT
    - T1
    - T2

I/O Module instance slot3

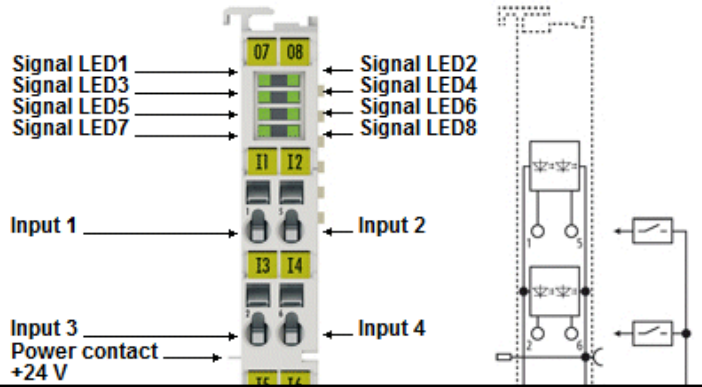
Digital input terminal to read the status of the SSI encoders of all 8 cover panels

System properties

Satisfies	cover_sys:panelDesign.requirements.absFeedbackStatus

Module type summary

ID	EL1088
Manufacturer	Beckhoff Automation
Description	8-channel digital input terminal 24V DC, negative switching
Used in	Cover (1), M3 (1)



Electrical design

OntoManager @ Merc

localhost:8080/elec?show=IoModuleInstance;qname=cover_elec:slot3

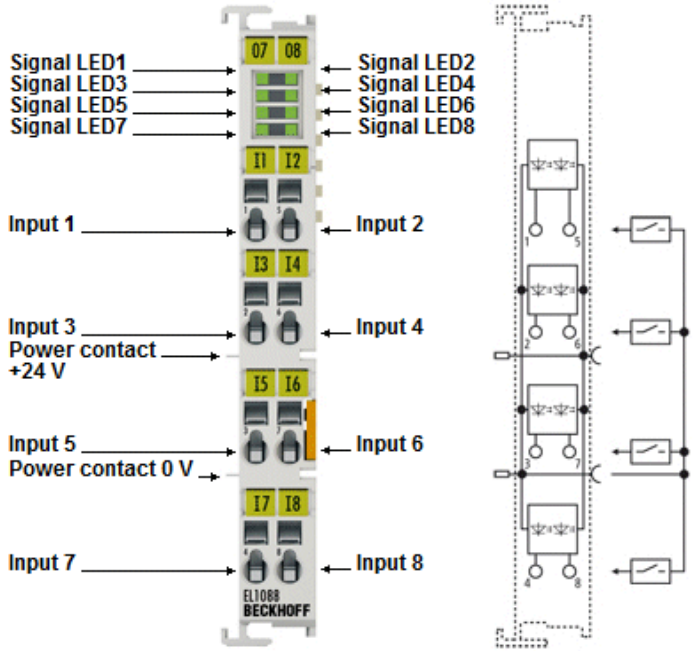
- slot3
- slot4
- slot5
- slot6
- slot7
- slot8
- slot9
- slot10
- slot11
- slot12
- slot13
- terminals
  - PE
  - L
  - N
  - 24V
  - GND
- connectors
  - ECAT
  - T1
  - T2
  - T3
  - T4
  - B1
  - B2
  - B3
  - B4
- M1
- M3
- Telemetry

System properties

Satisfies	cover_sys:panelDesign.requirements.absFeedbackStatus

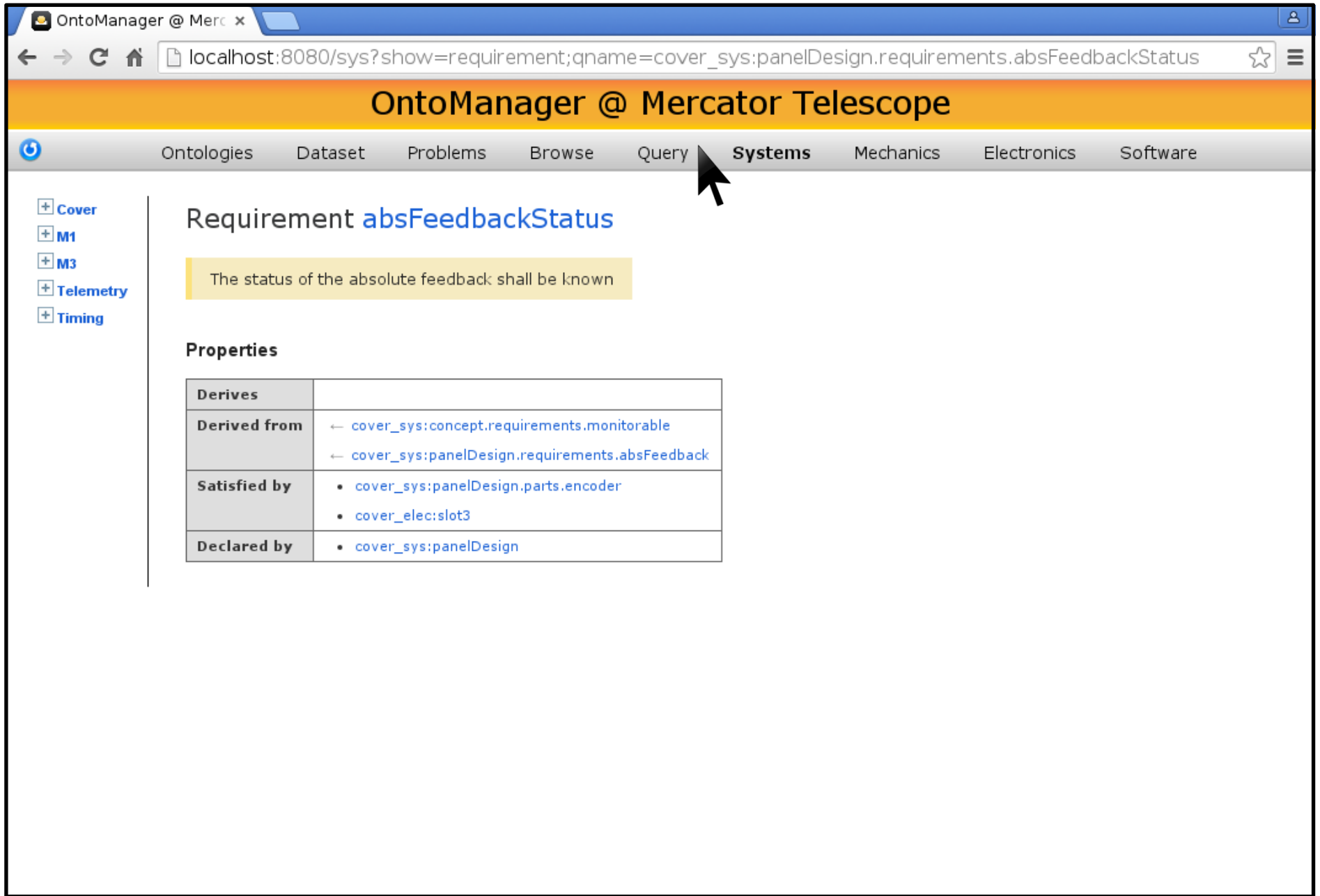
Module type summary

ID	EL1088
Manufacturer	Beckhoff Automation
Description	8-channel digital input terminal 24V DC, negative switching
Used in	Cover (1), M3 (1)

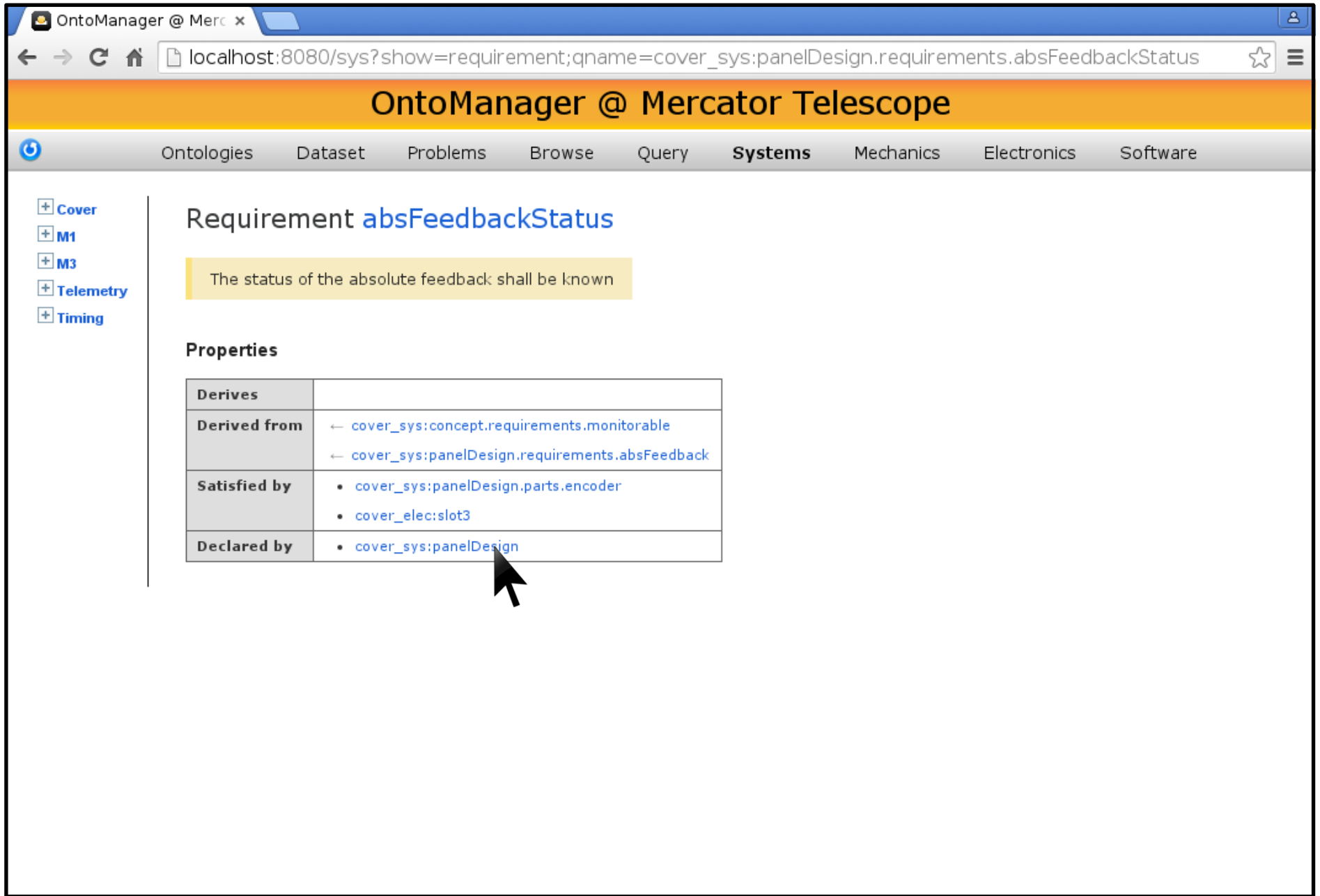


Connections

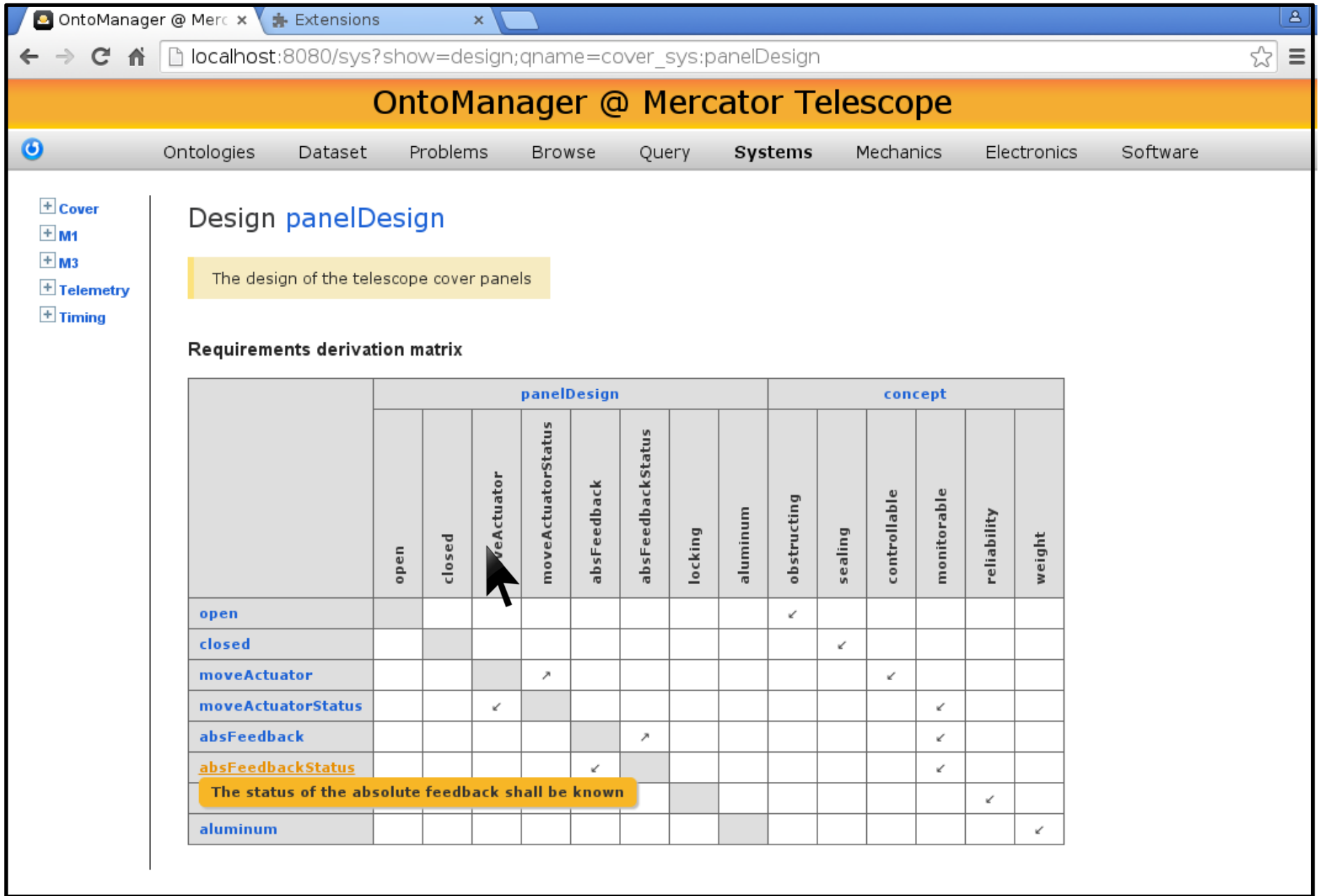
Systems design



Systems design



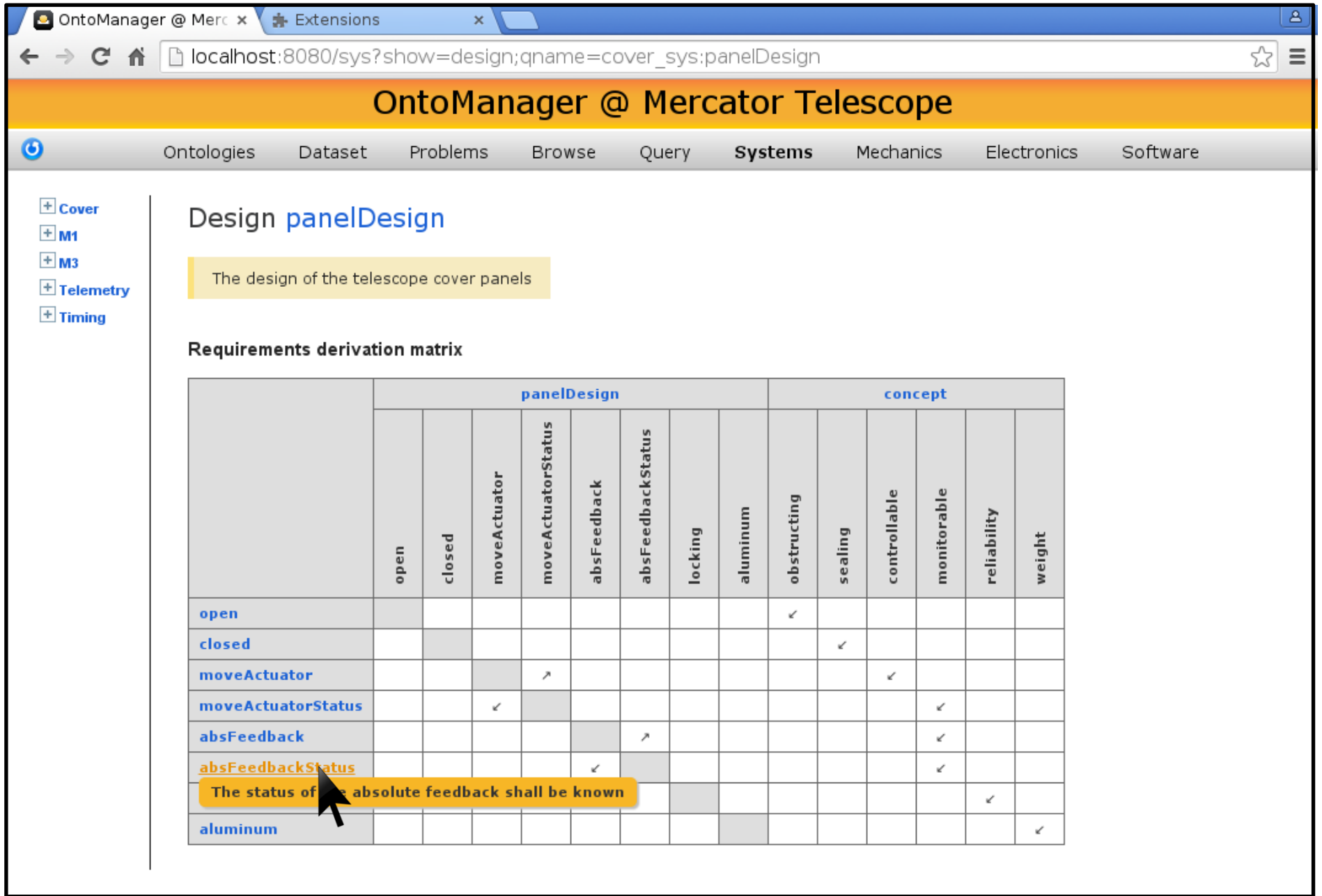
Systems design



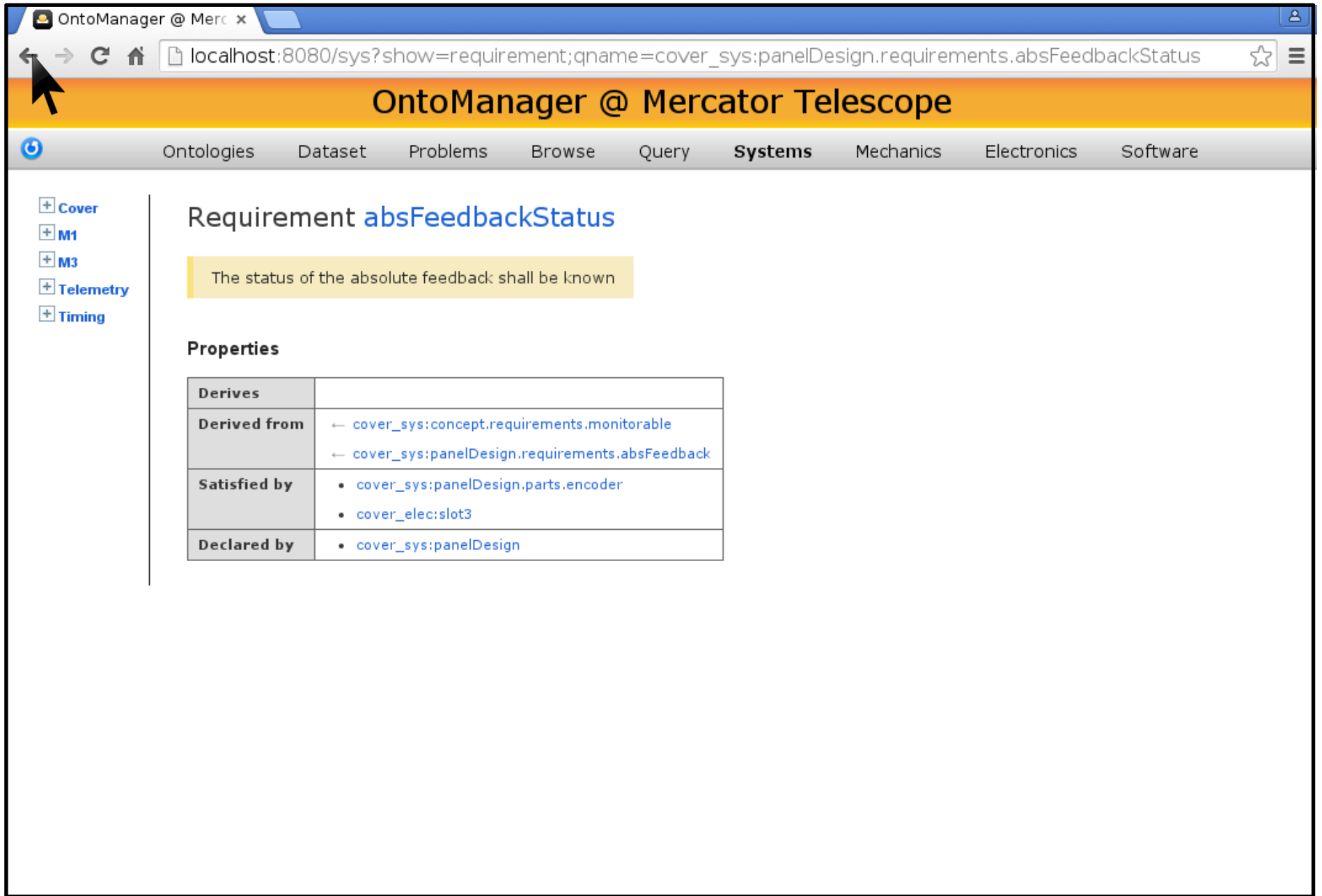
Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Systems design



Systems design



Electrical design

OntoManager @ Merc ×

localhost:8080/elec?show=IoModuleInstance;qname=cover_elec:slot3

OntoManager @ Mercator Telescope

Ontologies | Dataset | Problems | Browse | Query | Systems | Mechanics | **Electronics** | Software

- Cover
  - *I/O modules*
    - slot0
    - slot1
    - slot2
    - slot3
    - slot4
    - slot5
    - slot6
    - slot7
    - slot8
    - slot9
    - slot10
    - slot11
    - slot12
    - slot13
  - *terminals*
    - PE
    - L
    - N
    - 24V
    - GND
  - *connectors*
    - ECAT
    - T1
    - T2

I/O Module instance slot3

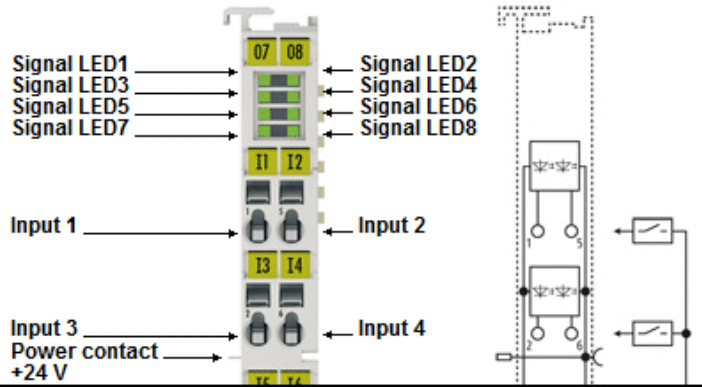
Digital input terminal to read the status of the SSI encoders of all 8 cover panels

System properties

Satisfies	cover_sys:panelDesign.requirements.absFeedbackStatus

Module type summary

ID	EL1088
Manufacturer	Beckhoff Automation
Description	8-channel digital input terminal 24V DC, negative switching
Used in	Cover (1), M3 (1)



Electrical design

OntoManager @ Merc ×

localhost:8080/elec?show=IoModuleInstance;qname=cover_elec:slot3

OntoManager @ Mercator Telescope

Ontologies | Dataset | Problems | Browse | Query | Systems | Mechanics | **Electronics** | Software

- Cover
  - *I/O modules*
    - slot0
    - slot1
    - slot2
    - slot3
    - slot4
    - slot5
    - slot6
    - slot7
    - slot8
    - slot9
    - slot10
    - slot11
    - slot12
    - slot13
  - *terminals*
    - PE
    - L
    - N
    - 24V
    - GND
  - *connectors*
    - ECAT
    - T1
    - T2

I/O Module instance slot3

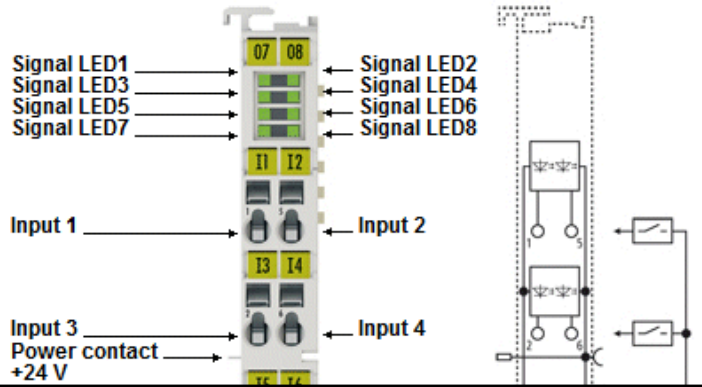
Digital input terminal to read the status of the SSI encoders of all 8 cover panels

System properties

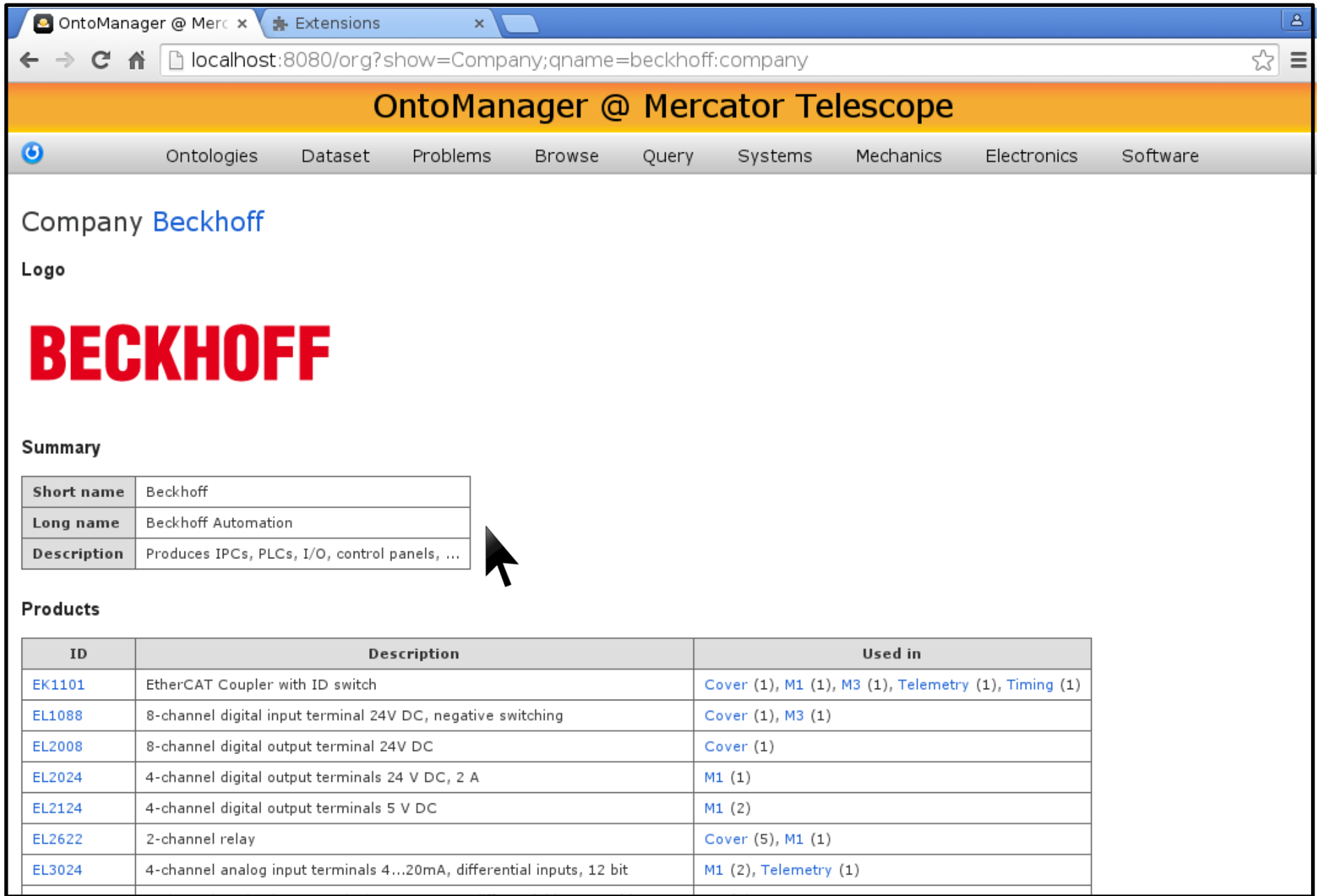
Satisfies	cover_sys:panelDesign.requirements.absFeedbackStatus

Module type summary

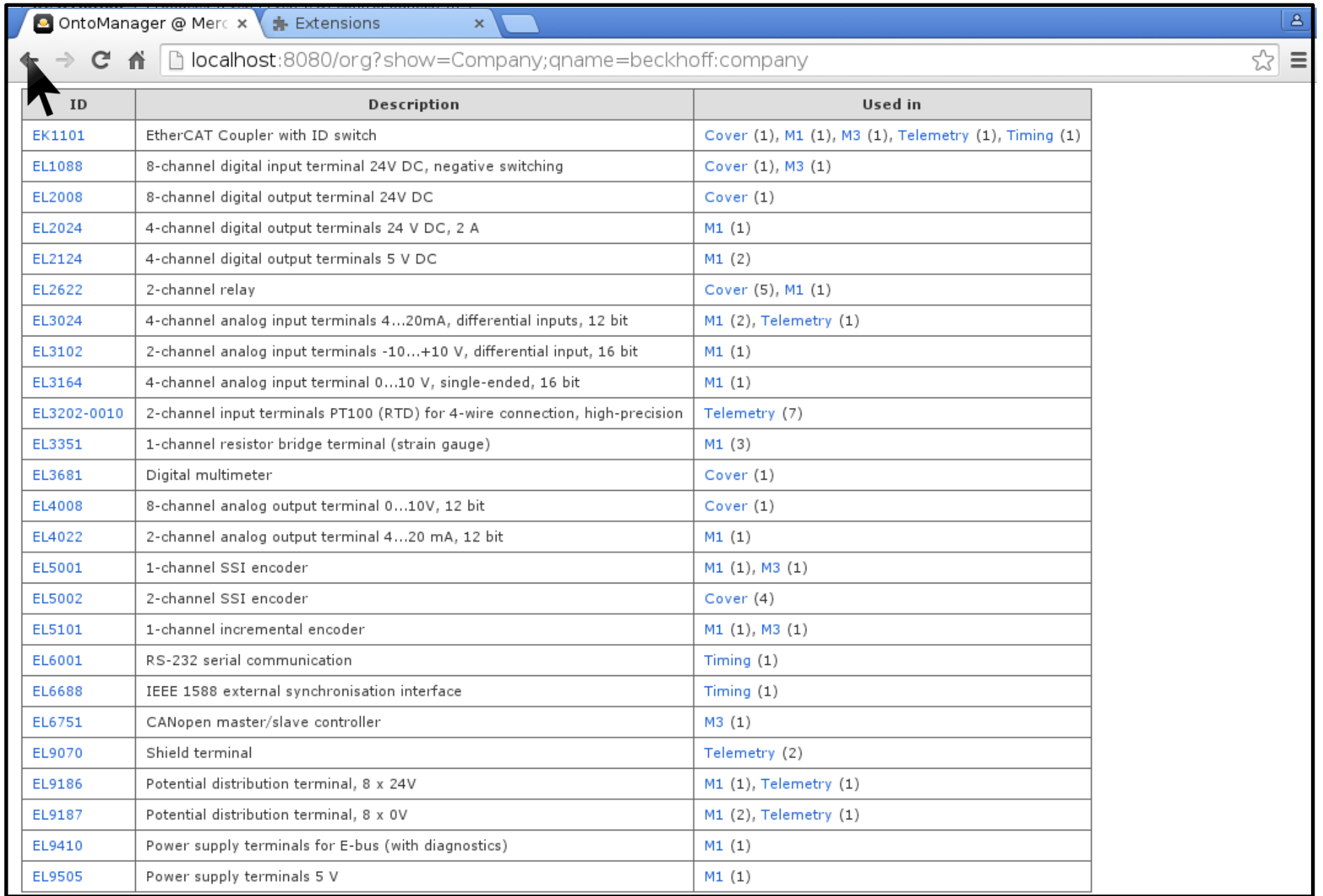
ID	EL1088
Manufacturer	Beckhoff Automation
Description	8-channel digital in terminal 24V DC, negative switching
Used in	Cover (1), M3 (1)



Electrical design

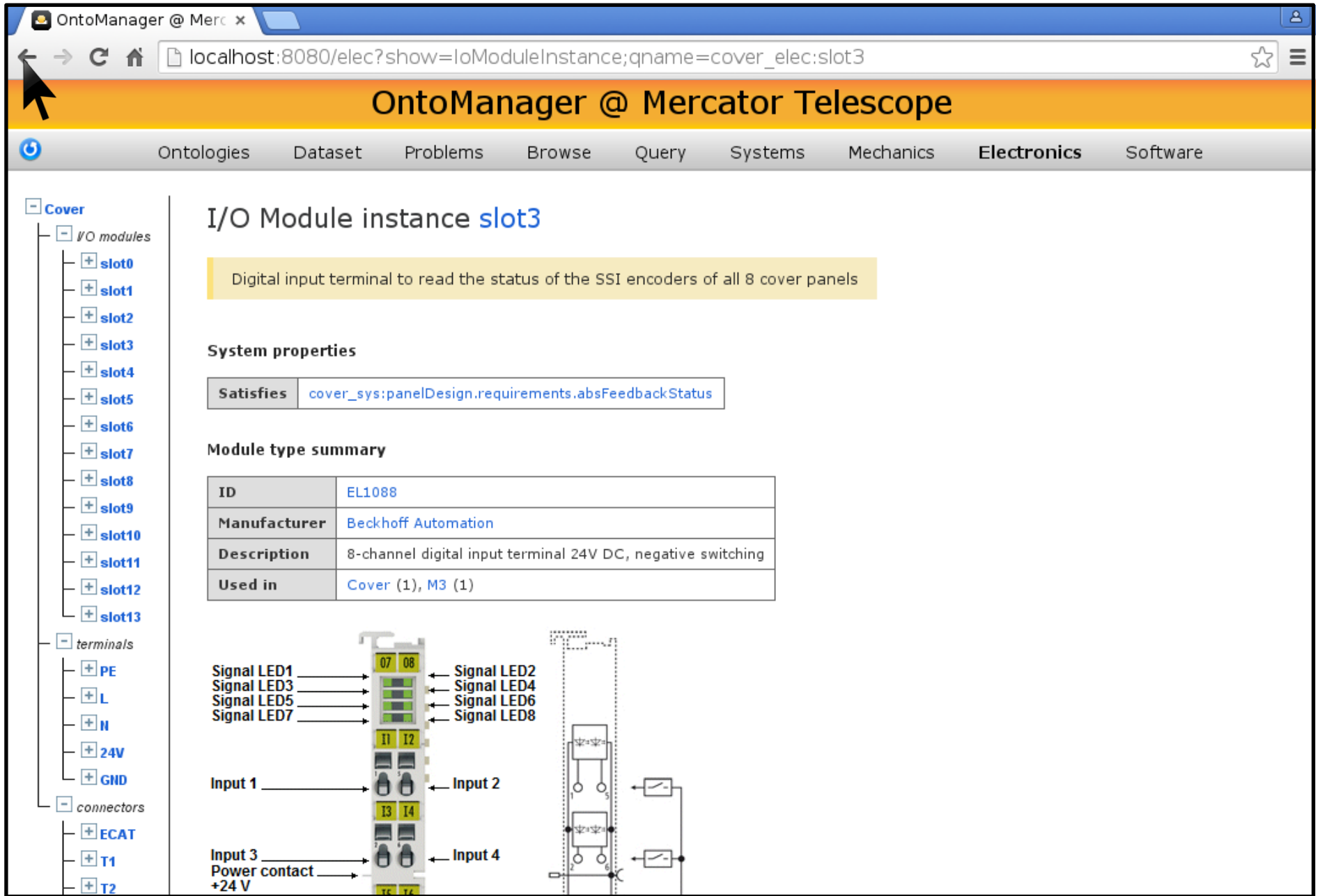


Electrical design

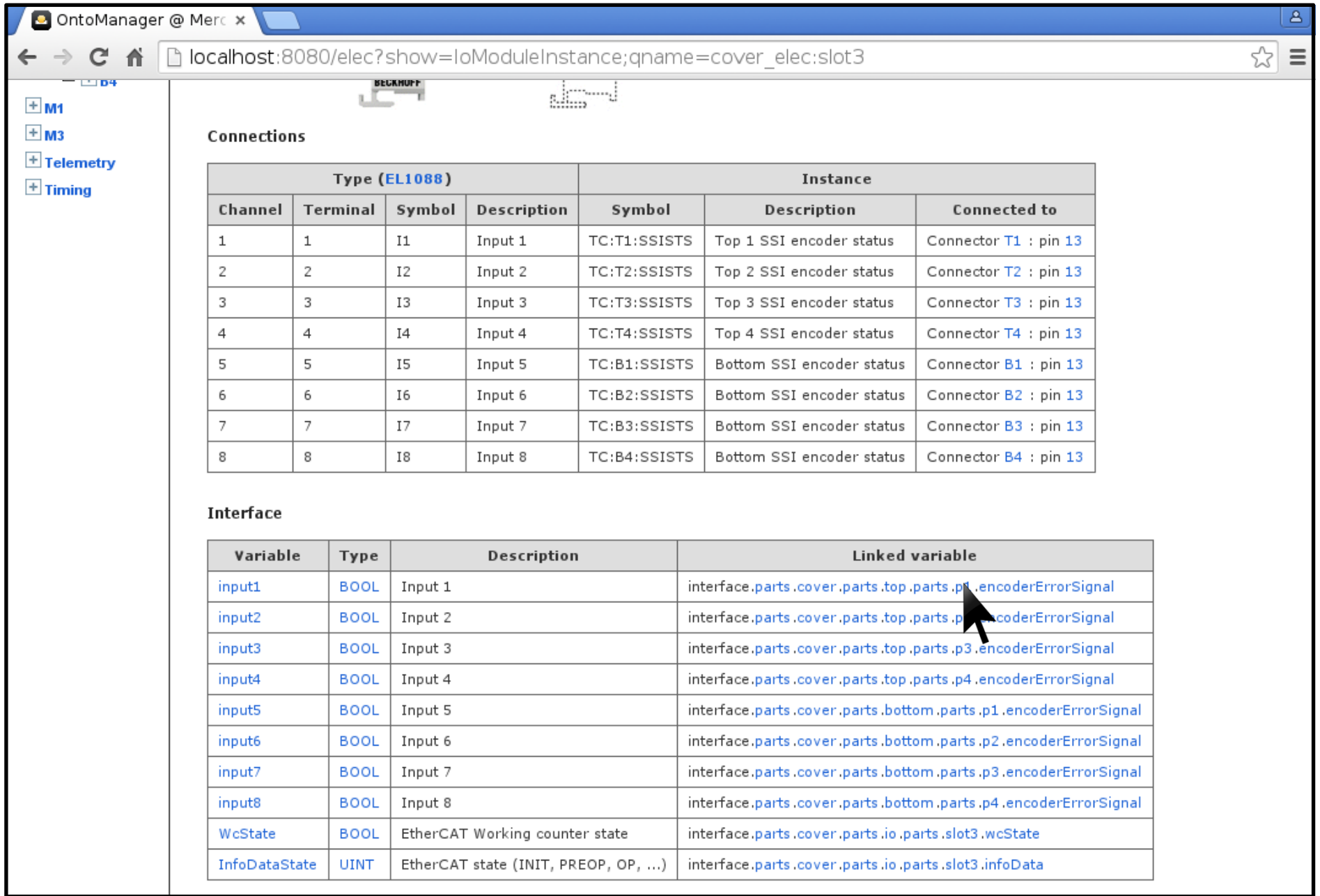


ID	Description	Used in
EK1101	EtherCAT Coupler with ID switch	Cover (1), M1 (1), M3 (1), Telemetry (1), Timing (1)
EL1088	8-channel digital input terminal 24V DC, negative switching	Cover (1), M3 (1)
EL2008	8-channel digital output terminal 24V DC	Cover (1)
EL2024	4-channel digital output terminals 24 V DC, 2 A	M1 (1)
EL2124	4-channel digital output terminals 5 V DC	M1 (2)
EL2622	2-channel relay	Cover (5), M1 (1)
EL3024	4-channel analog input terminals 4...20mA, differential inputs, 12 bit	M1 (2), Telemetry (1)
EL3102	2-channel analog input terminals -10...+10 V, differential input, 16 bit	M1 (1)
EL3164	4-channel analog input terminal 0...10 V, single-ended, 16 bit	M1 (1)
EL3202-0010	2-channel input terminals PT100 (RTD) for 4-wire connection, high-precision	Telemetry (7)
EL3351	1-channel resistor bridge terminal (strain gauge)	M1 (3)
EL3681	Digital multimeter	Cover (1)
EL4008	8-channel analog output terminal 0...10V, 12 bit	Cover (1)
EL4022	2-channel analog output terminal 4...20 mA, 12 bit	M1 (1)
EL5001	1-channel SSI encoder	M1 (1), M3 (1)
EL5002	2-channel SSI encoder	Cover (4)
EL5101	1-channel incremental encoder	M1 (1), M3 (1)
EL6001	RS-232 serial communication	Timing (1)
EL6688	IEEE 1588 external synchronisation interface	Timing (1)
EL6751	CANopen master/slave controller	M3 (1)
EL9070	Shield terminal	Telemetry (2)
EL9186	Potential distribution terminal, 8 x 24V	M1 (1), Telemetry (1)
EL9187	Potential distribution terminal, 8 x 0V	M1 (2), Telemetry (1)
EL9410	Power supply terminals for E-bus (with diagnostics)	M1 (1)
EL9505	Power supply terminals 5 V	M1 (1)

Electrical design



Electrical design



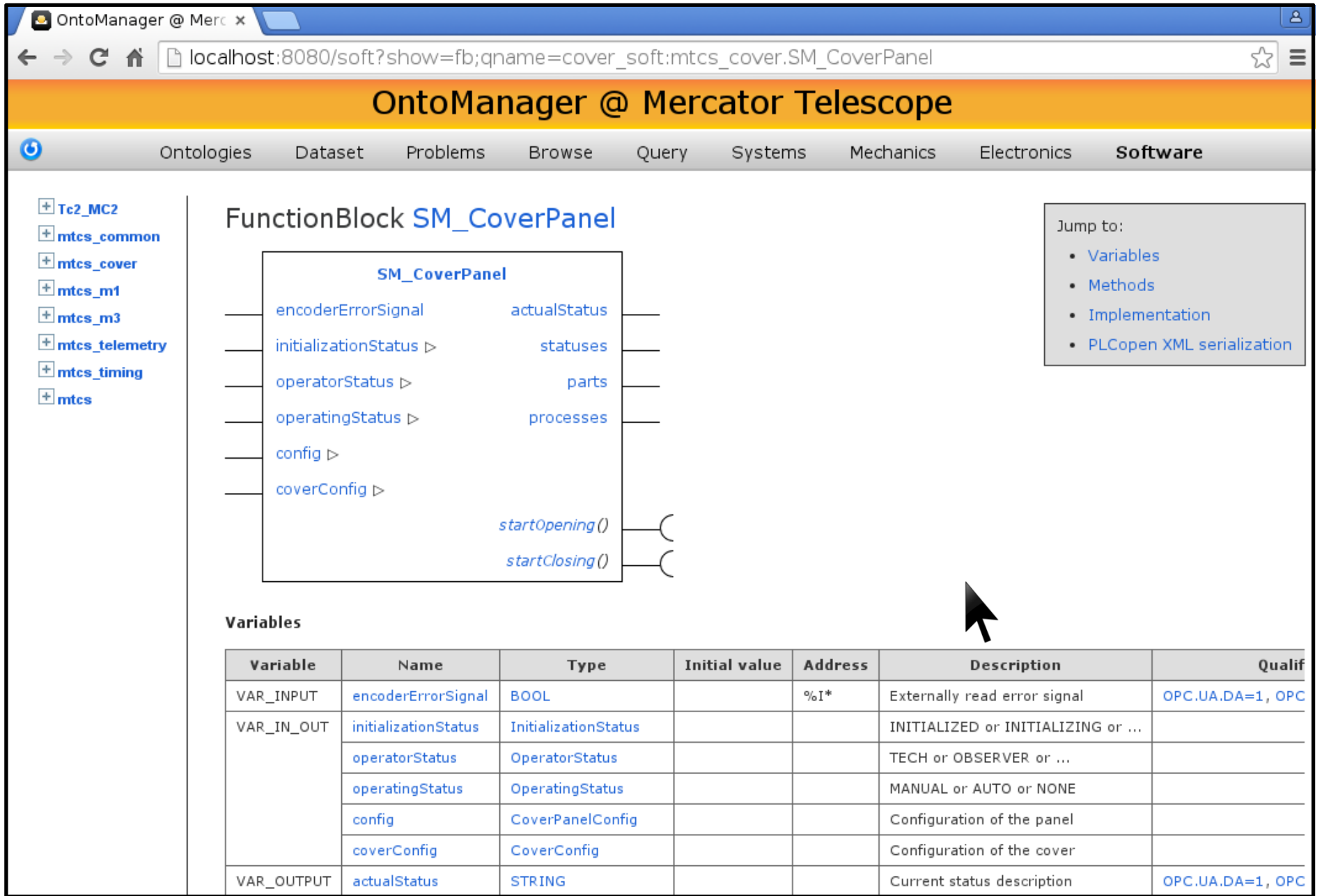
Connections

Type (EL1088)				Instance		
Channel	Terminal	Symbol	Description	Symbol	Description	Connected to
1	1	I1	Input 1	TC:T1:SSISTS	Top 1 SSI encoder status	Connector T1 : pin 13
2	2	I2	Input 2	TC:T2:SSISTS	Top 2 SSI encoder status	Connector T2 : pin 13
3	3	I3	Input 3	TC:T3:SSISTS	Top 3 SSI encoder status	Connector T3 : pin 13
4	4	I4	Input 4	TC:T4:SSISTS	Top 4 SSI encoder status	Connector T4 : pin 13
5	5	I5	Input 5	TC:B1:SSISTS	Bottom SSI encoder status	Connector B1 : pin 13
6	6	I6	Input 6	TC:B2:SSISTS	Bottom SSI encoder status	Connector B2 : pin 13
7	7	I7	Input 7	TC:B3:SSISTS	Bottom SSI encoder status	Connector B3 : pin 13
8	8	I8	Input 8	TC:B4:SSISTS	Bottom SSI encoder status	Connector B4 : pin 13

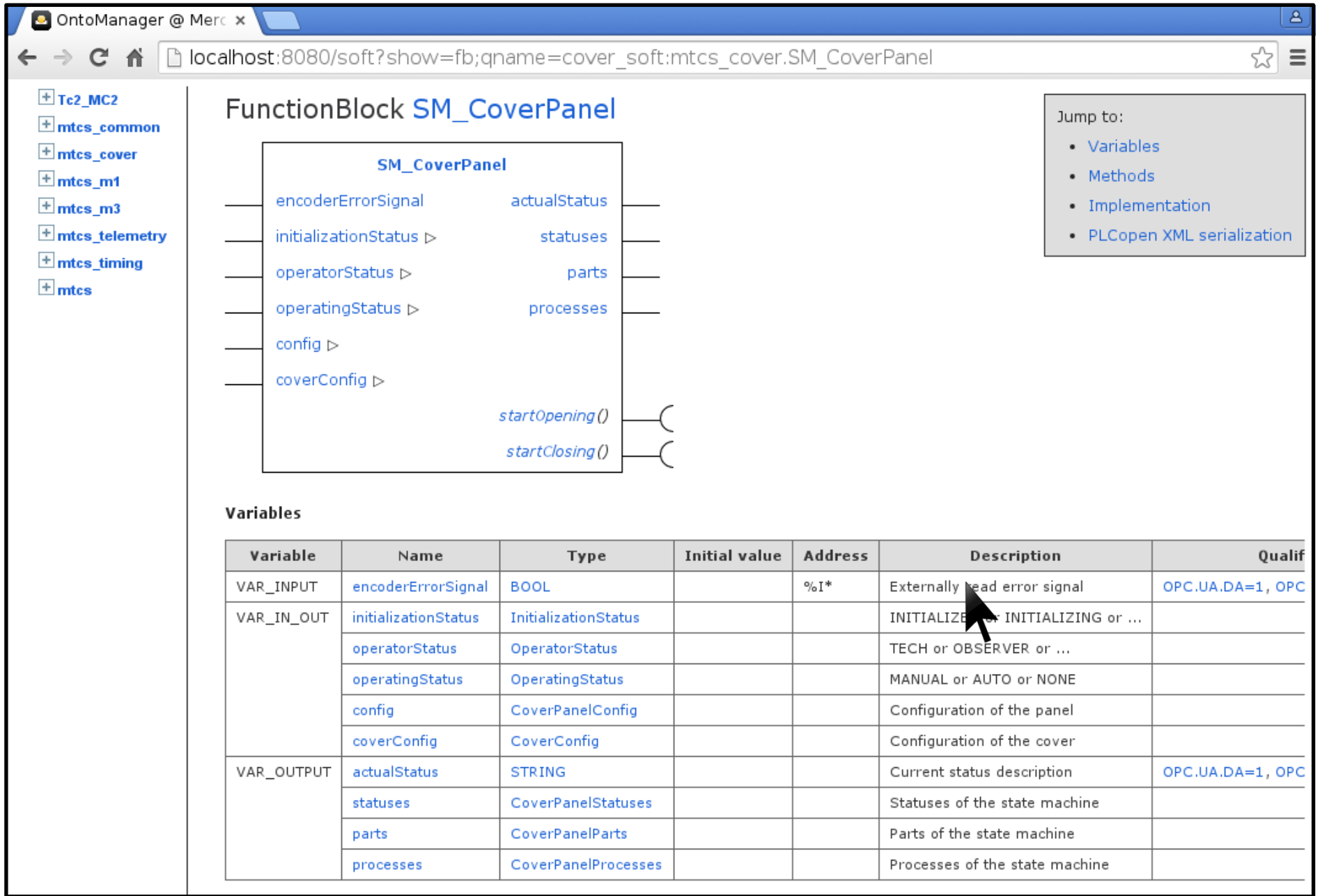
Interface

Variable	Type	Description	Linked variable
input1	BOOL	Input 1	interface.parts.cover.parts.top.parts.p1.encoderErrorSignal
input2	BOOL	Input 2	interface.parts.cover.parts.top.parts.p2.encoderErrorSignal
input3	BOOL	Input 3	interface.parts.cover.parts.top.parts.p3.encoderErrorSignal
input4	BOOL	Input 4	interface.parts.cover.parts.top.parts.p4.encoderErrorSignal
input5	BOOL	Input 5	interface.parts.cover.parts.bottom.parts.p1.encoderErrorSignal
input6	BOOL	Input 6	interface.parts.cover.parts.bottom.parts.p2.encoderErrorSignal
input7	BOOL	Input 7	interface.parts.cover.parts.bottom.parts.p3.encoderErrorSignal
input8	BOOL	Input 8	interface.parts.cover.parts.bottom.parts.p4.encoderErrorSignal
WcState	BOOL	EtherCAT Working counter state	interface.parts.cover.parts.io.parts.slot3.wcState
InfoDataState	UINT	EtherCAT state (INIT, PREOP, OP, ...)	interface.parts.cover.parts.io.parts.slot3.infoData

Software design



Software design



Software design

OntoManager @ Merc ×

localhost:8080/soft?show=fb;qname=cover_soft:mtcs_cover.SM_CoverPanel

VAR_INPUT	encoderErrorSignal	BOOL		%I*	Externally read error signal	OPC.UA.DA=1, OPC
VAR_IN_OUT	initializationStatus	InitializationStatus			INITIALIZED or INITIALIZING or ...	
	operatorStatus	OperatorStatus			TECH or OBSERVER or ...	
	operatingStatus	OperatingStatus			MANUAL or AUTO or NONE	
	config	CoverPanelConfig			Configuration of the panel	
	coverConfig	CoverConfig			Configuration of the cover	
VAR_OUTPUT	actualStatus	STRING			Current status description	OPC.UA.DA=1, OPC
	statuses	CoverPanelStatuses			Statuses of the state machine	
	parts	CoverPanelParts			Parts of the state machine	
	processes	CoverPanelProcesses			Processes of the state machine	

Methods

- **startOpening**()

Comment	Start opening the panel						
Return type	RequestResults						
Interface	**Variable**	**Name**	**Type**	**Initial value**	**Address**	**Description**	**Qualifiers**
Implementation	`startOpening := THIS^.processes.startOpening.request();`						

- **startClosing**()

Comment	Start closing the panel						
Return type	RequestResults						
Interface	**Variable**	**Name**	**Type**	**Initial value**	**Address**	**Description**	**Qualifiers**
Implementation	`startClosing := THIS^.processes.startClosing.request();`						

Software design

OntoManager @ Merc ×

localhost:8080/soft?show=fb;qname=cover_soft:mtcs_cover.SM_CoverPanel

Implementation	startOpening := THIS^.processes.startOpening.request();

- **startClosing()**

Comment	Start closing the panel						
Return type	RequestResults						
Interface	Variable	Name	Type	Initial value	Address	Description	Qualifiers
Implementation	startClosing := THIS^.processes.startClosing.request();						

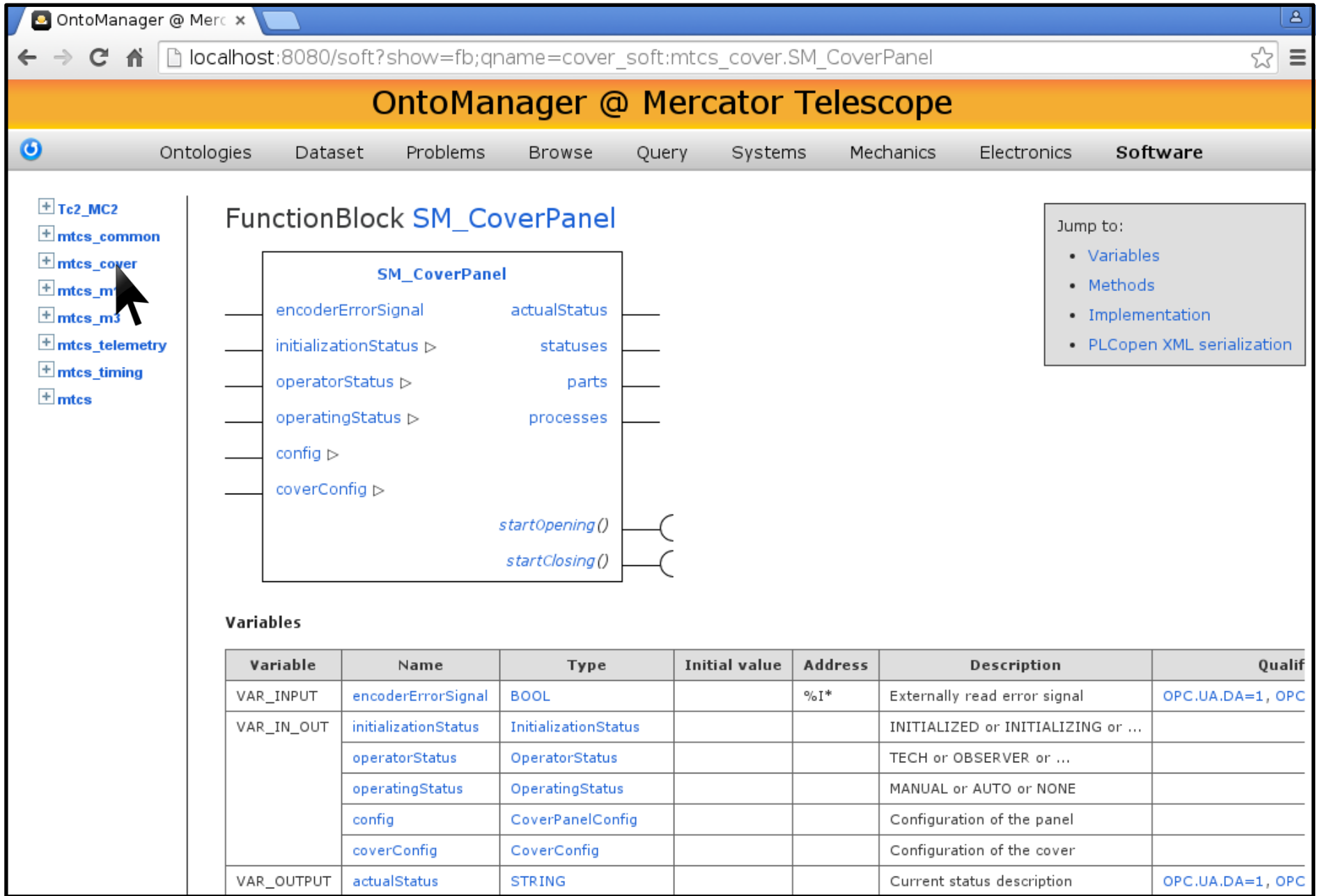
Implementation

```
parts.axis(
    isEnabled := operatorStatus.tech AND (operatingStatus.manual AND initializationStatus.initialized),
    standstillTolerance := config.standstillTolerance);
parts.motorRelay( isEnabled := parts.axis.isEnabled );
statuses.busyStatus( isBusy := parts.axis.statuses.busyStatus.busy OR parts.motorRelay.statuses.busyStatus.busy );
statuses.apertureStatus(
    isOpen := (ABS(config.openPosition - parts.axis.actPos.degrees.value)) < config.openTolerance,
    isClosed := (ABS(config.closedPosition - parts.axis.actPos.degrees.value)) < config.closedTolerance);
statuses.healthStatus(
    isGood := parts.axis.statuses.healthStatus.isGood AND (NOT(encoderErrorSignal)),
    hasWarning := parts.axis.statuses.healthStatus.hasWarning);
statuses.openingStatus(
    isOpening := parts.axis.statuses.motionStatus.backward,
    isClosing := parts.axis.statuses.motionStatus.forward);
processes.startOpening( isEnabled := operatorStatus.tech AND (operatingStatus.manual AND initializationStatus.initialized) )
processes.startClosing( isEnabled := operatorStatus.tech AND (operatingStatus.manual AND initializationStatus.initialized) )
```

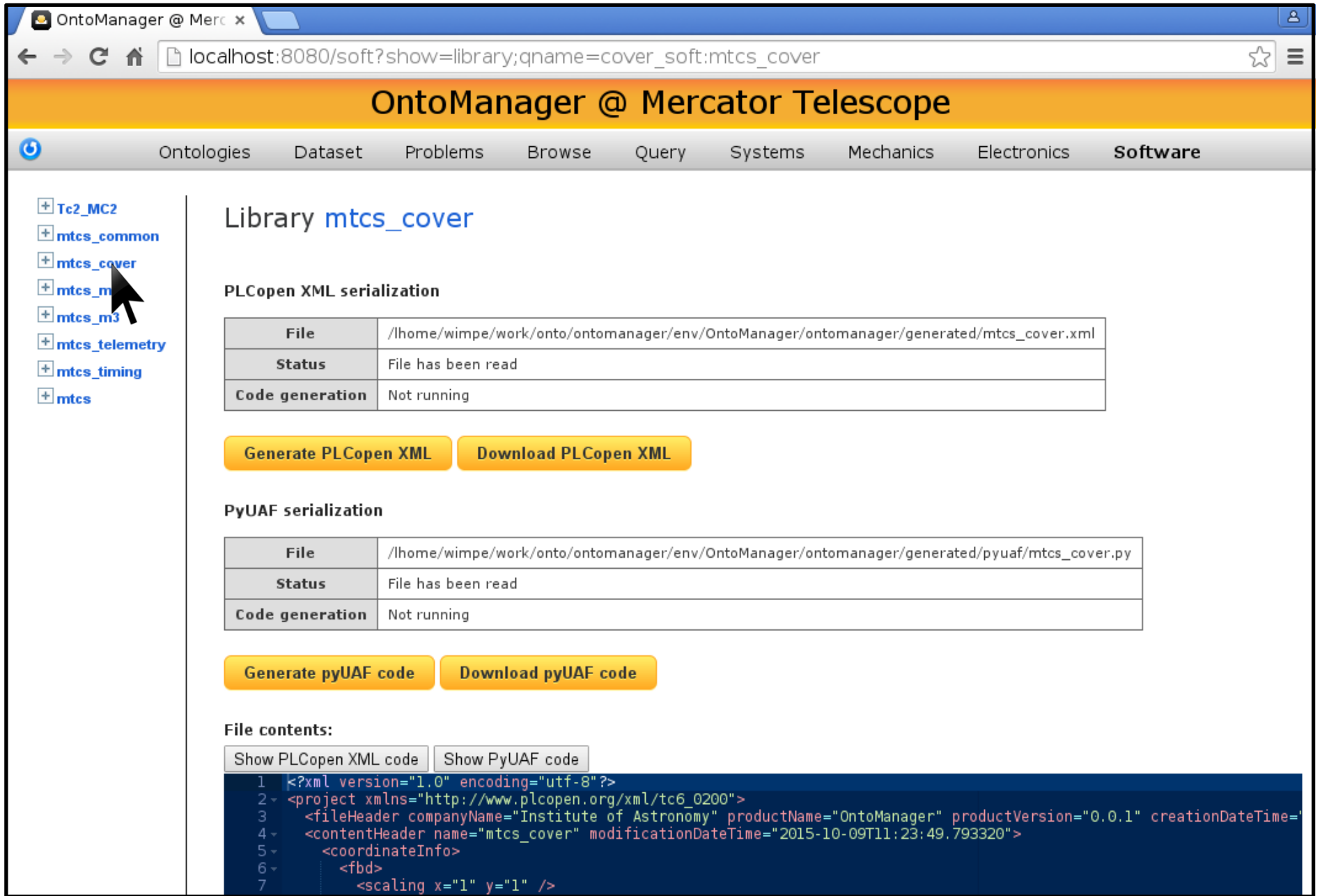
PLCopen XML serialization

```
1  <pou name="SM_CoverPanel" pouType="functionBlock">
2    <interface>
3      <inputVars>
4        <variable name="encoderErrorSignal" address="%I*">
5          <type><BOOL /></type>
6          <addData>
```

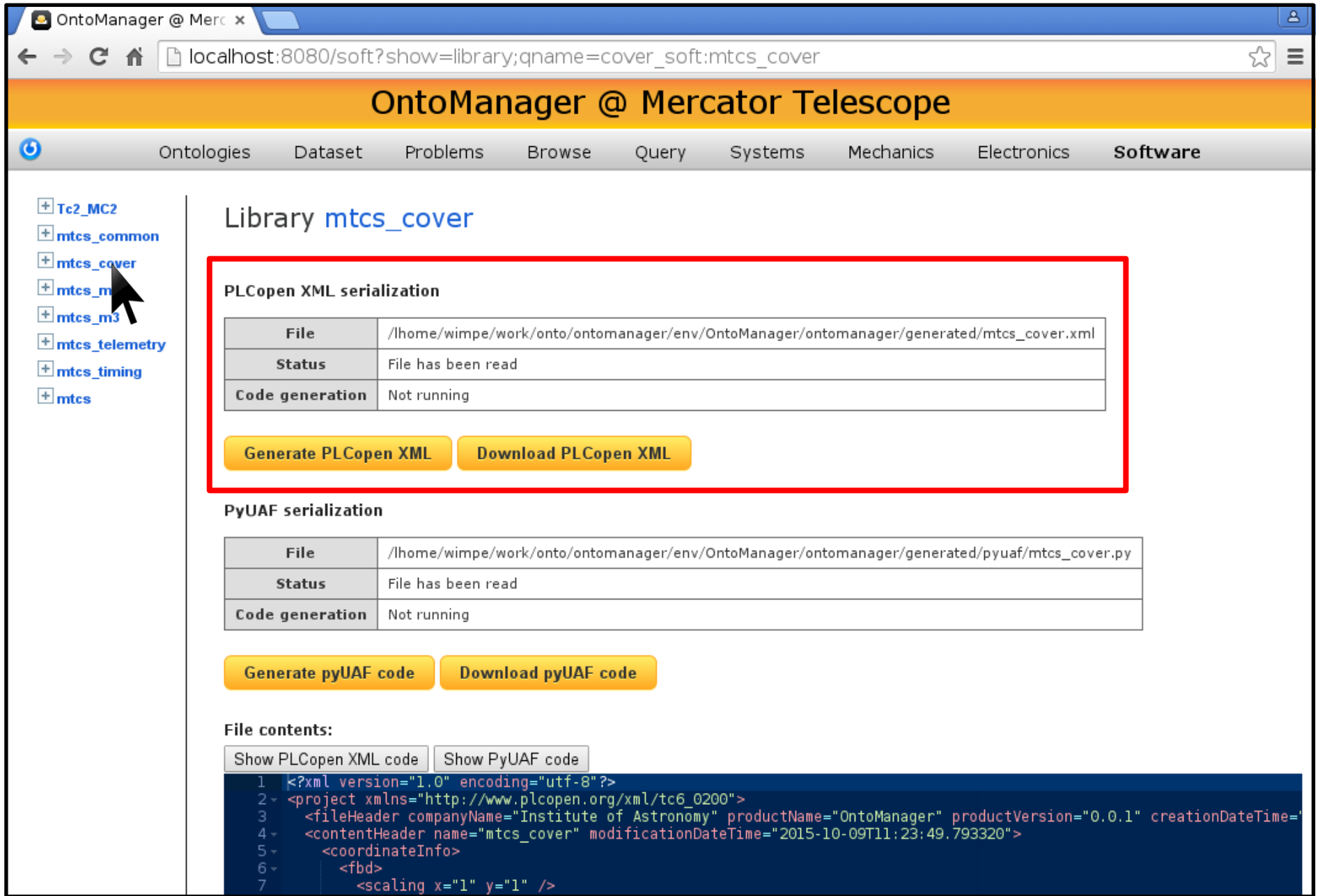
Software design



Software design



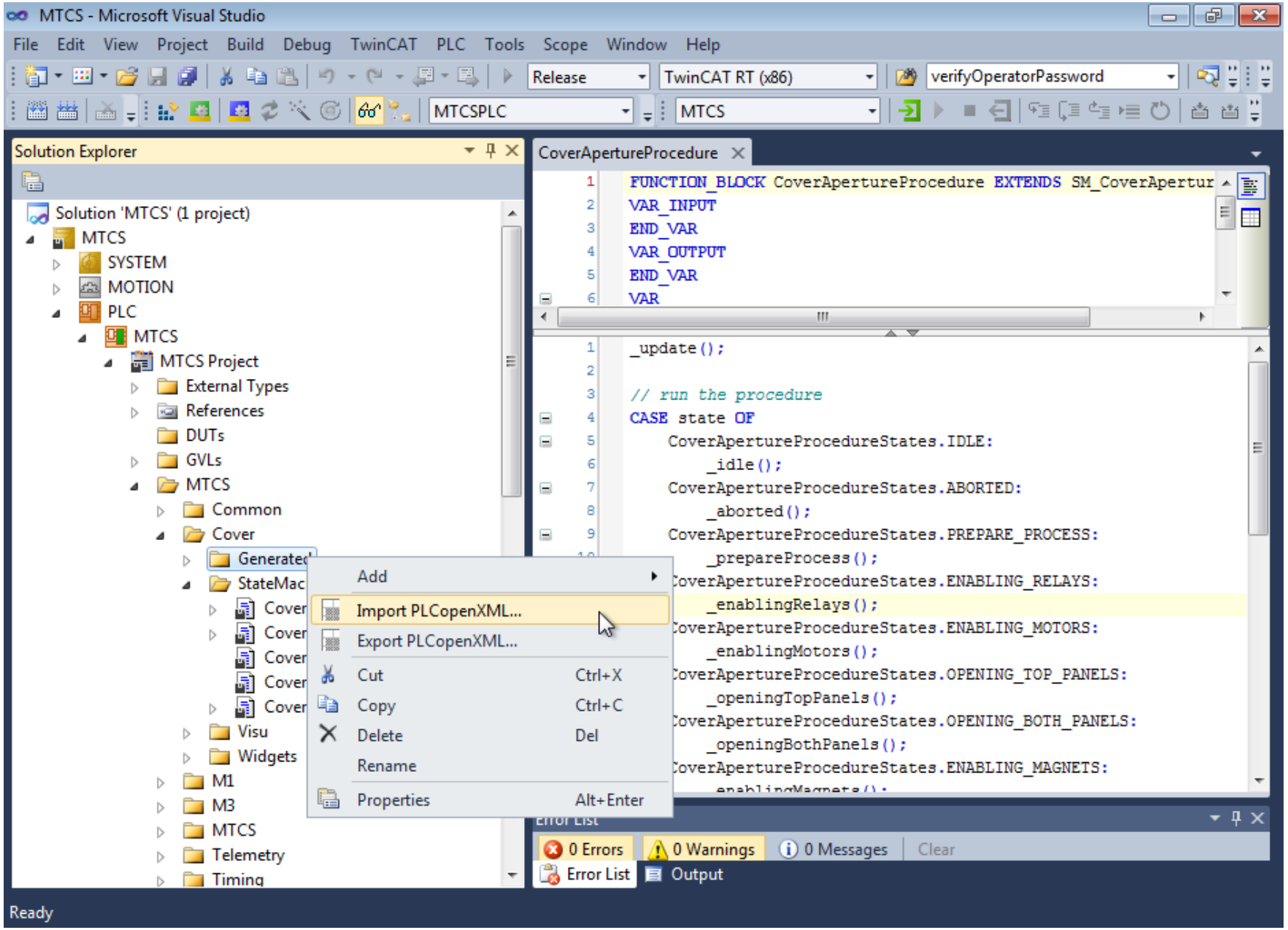
Software design



Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

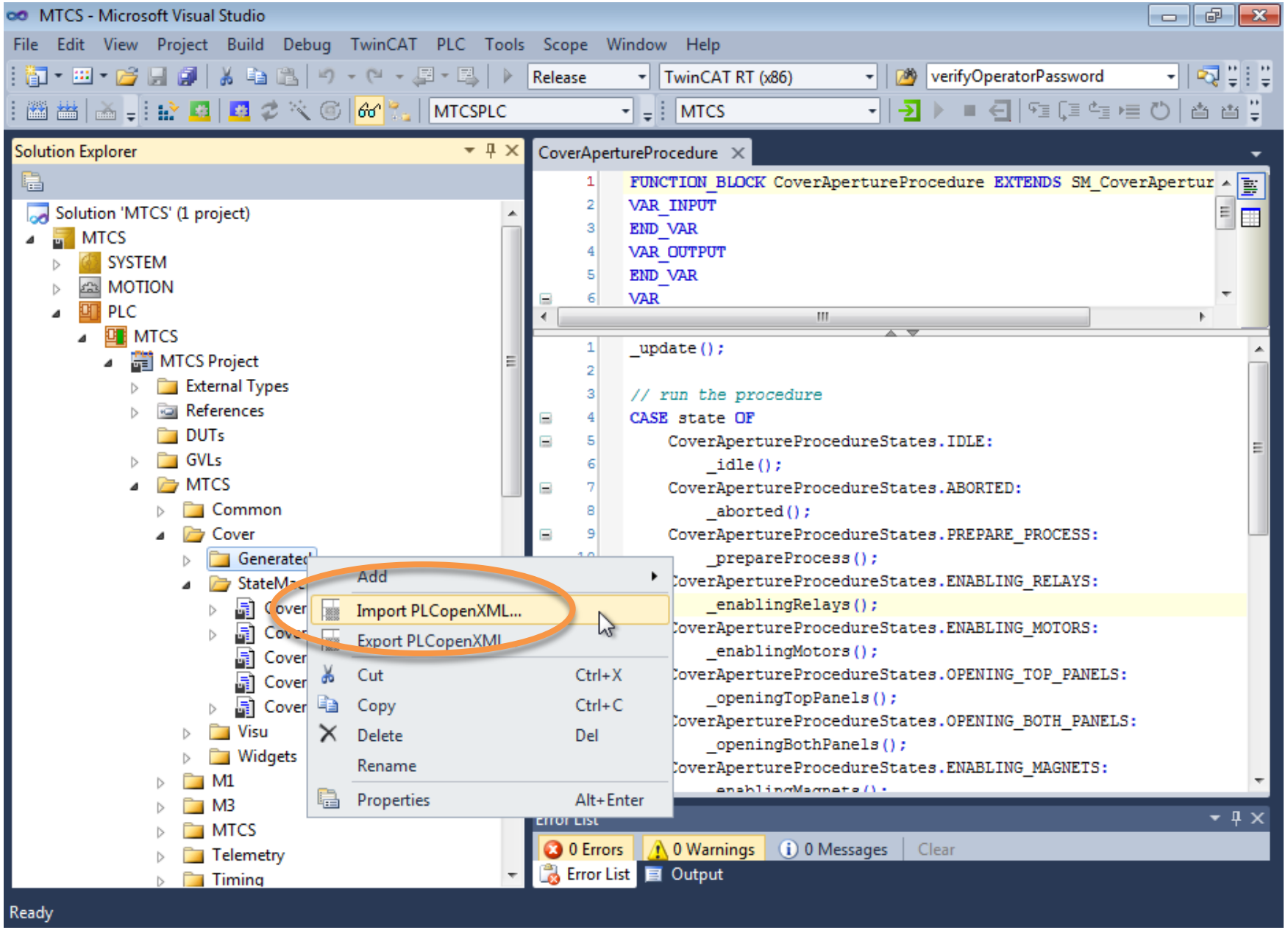
Software design



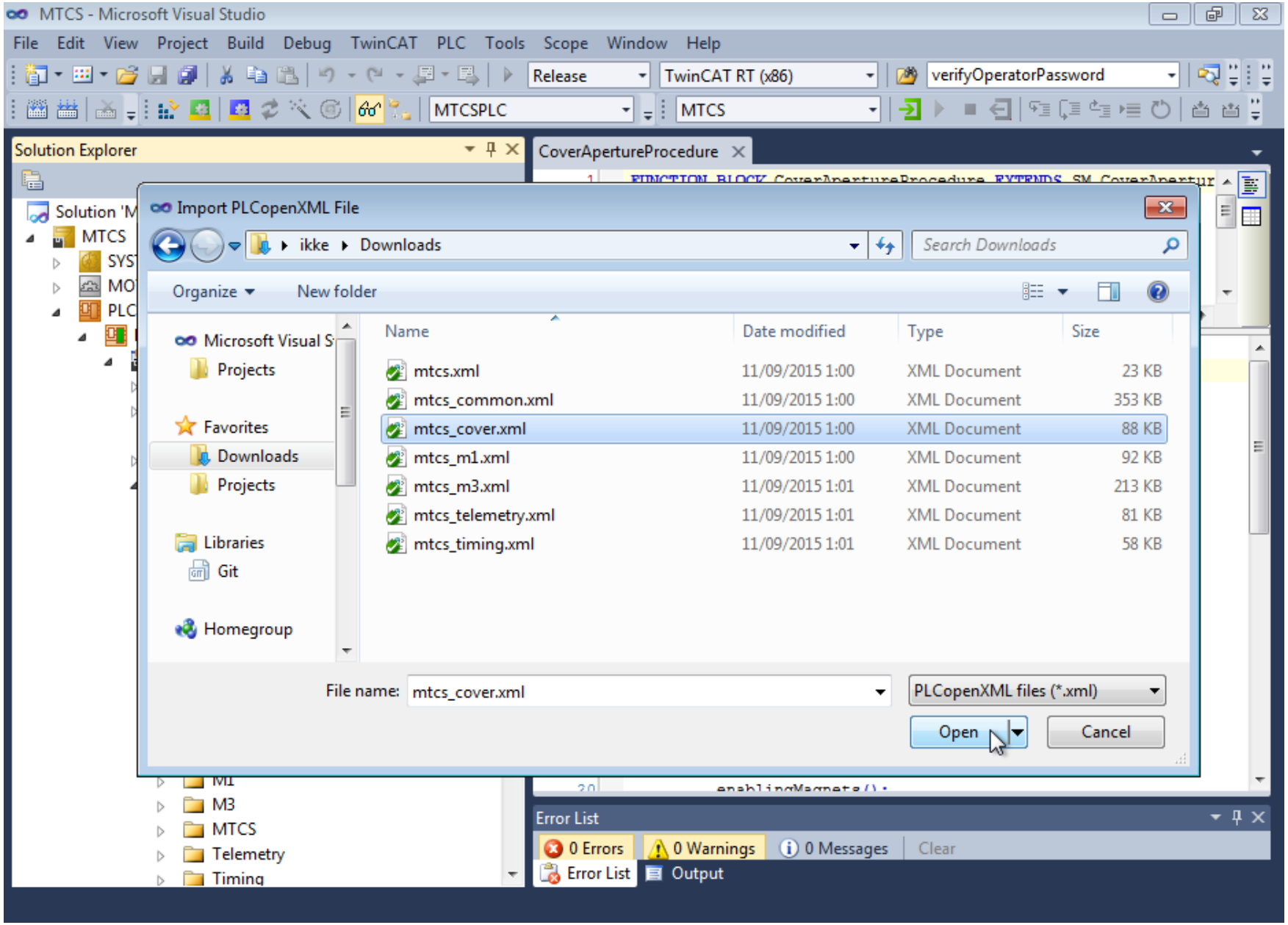
Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

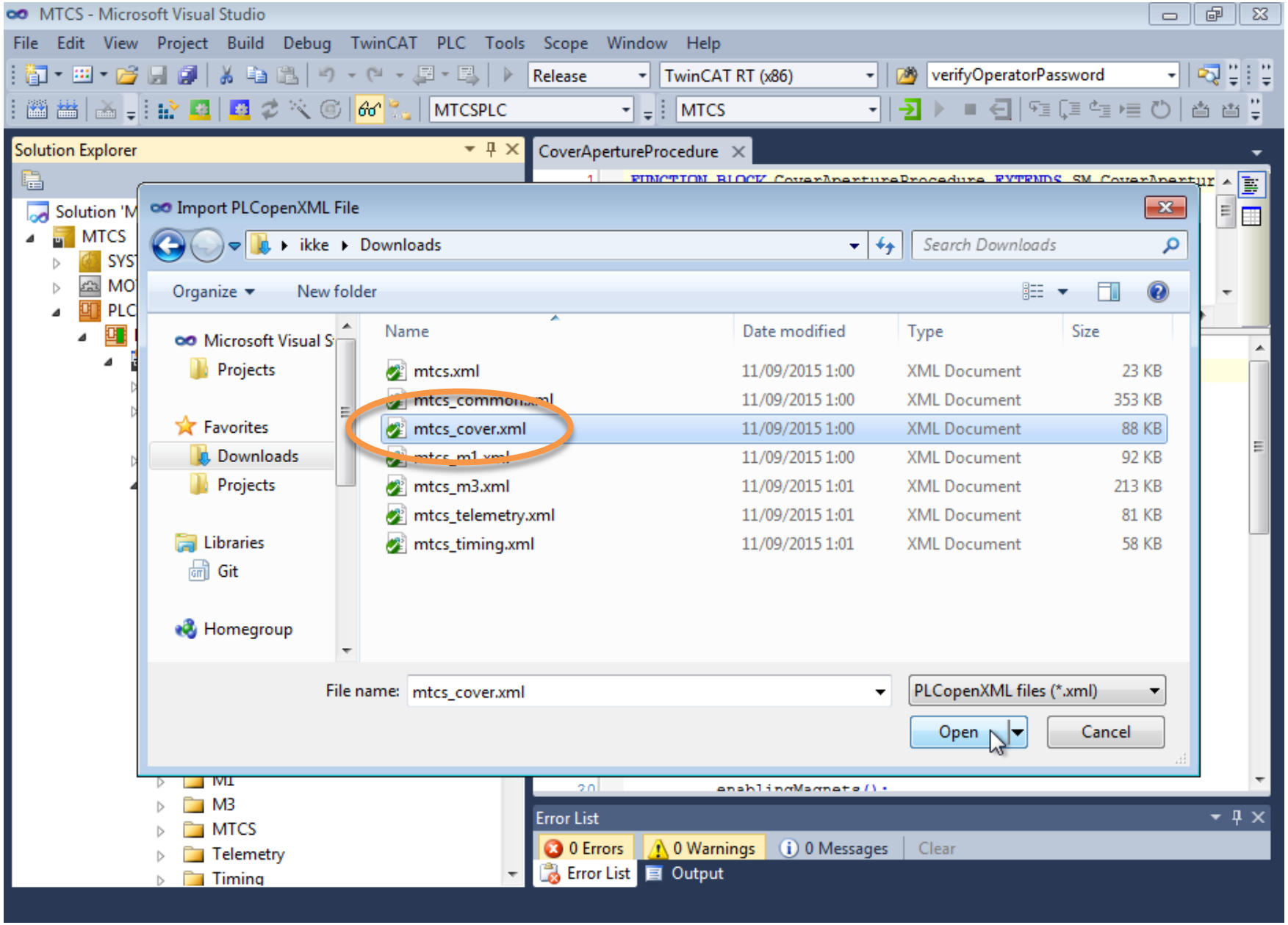
Software design



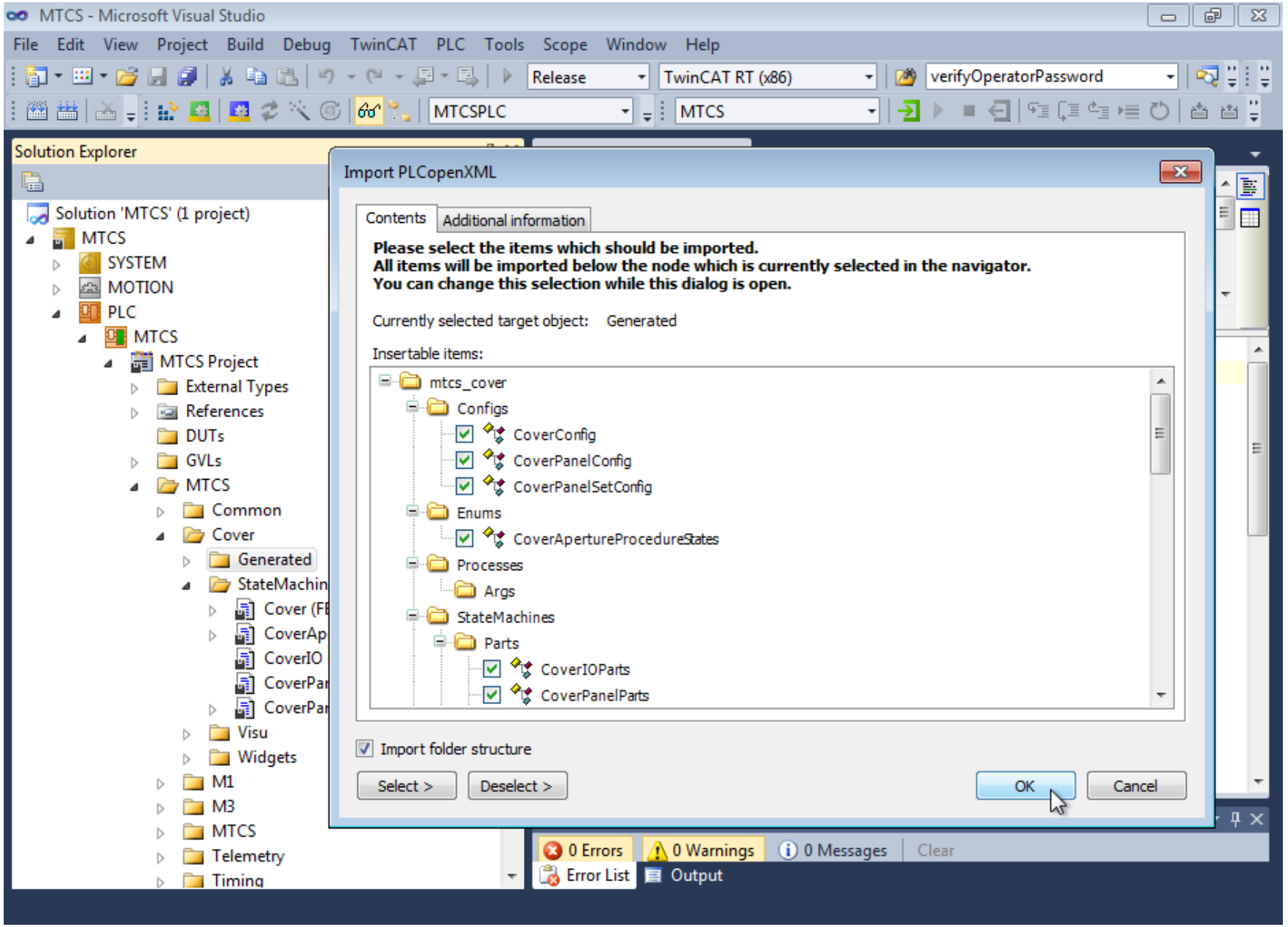
Software design



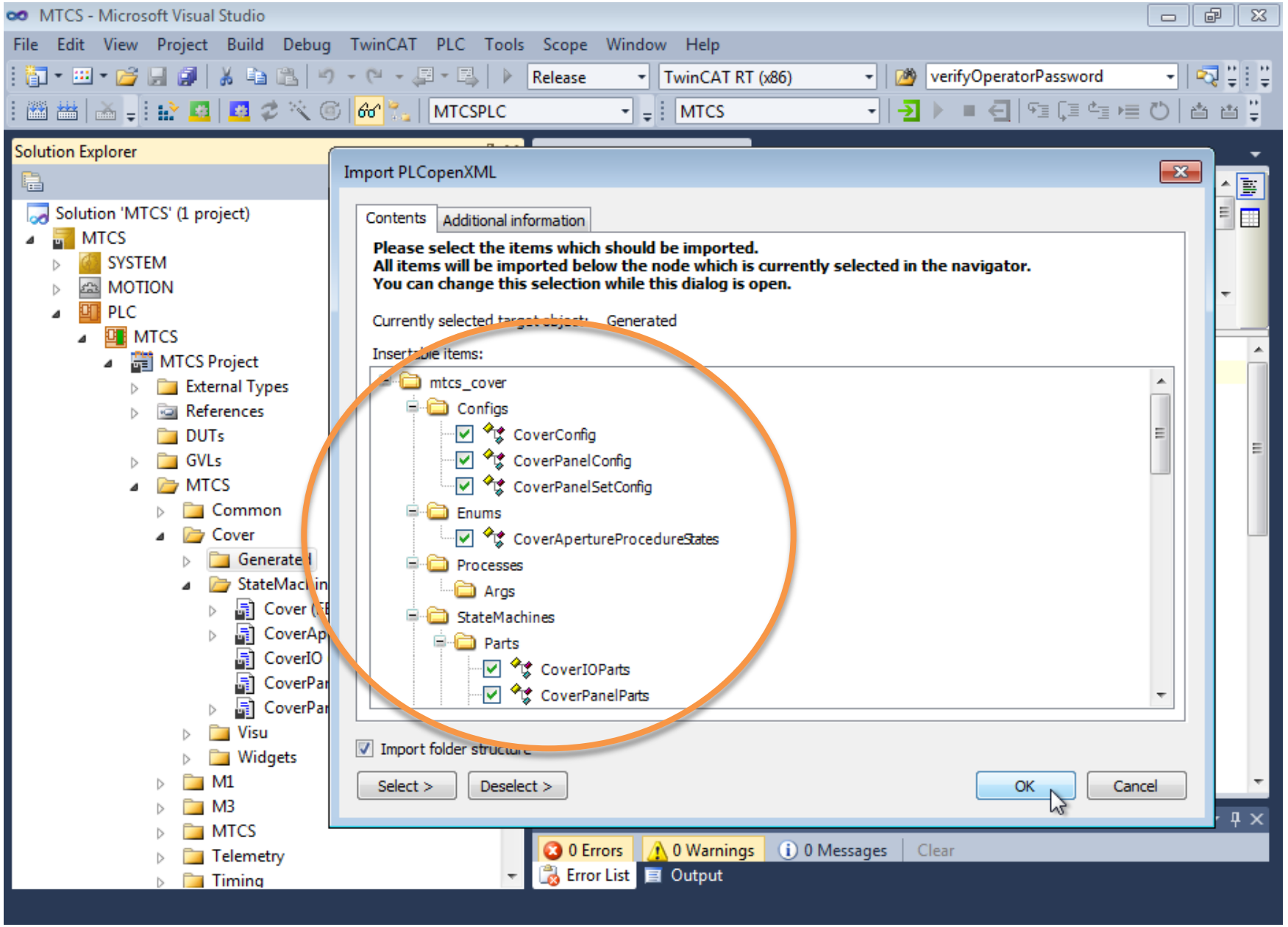
Software design



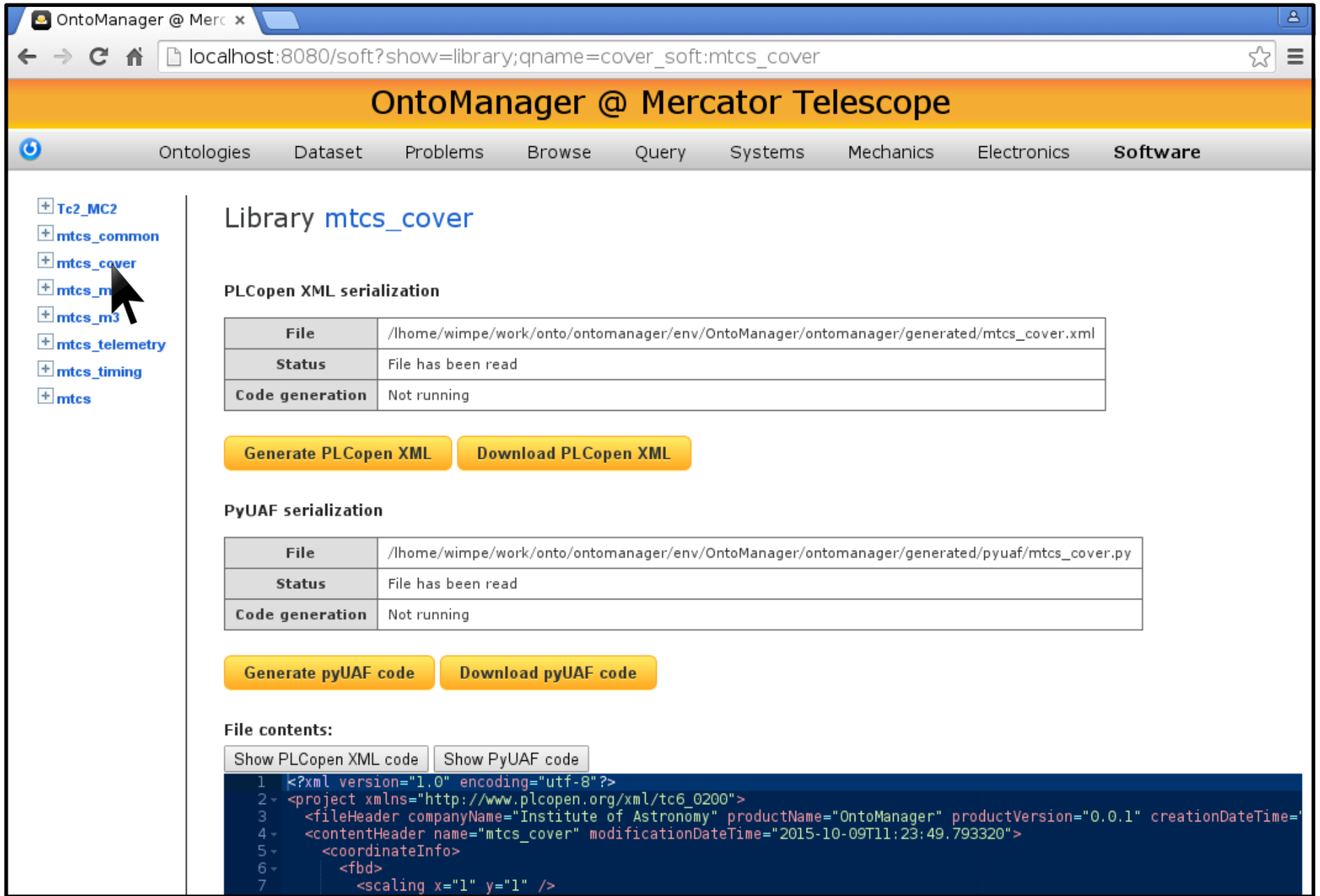
Software design



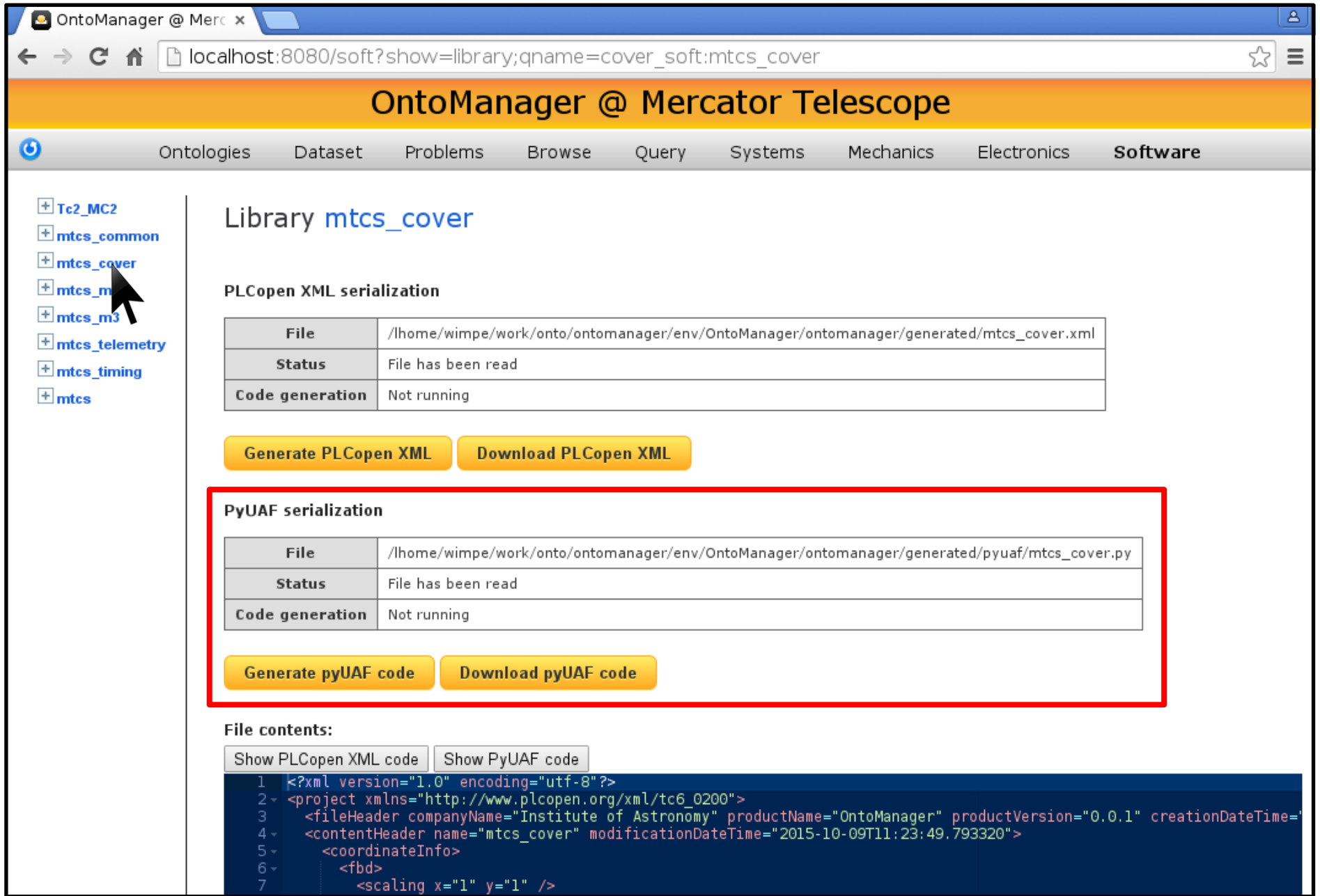
Software design



Software design

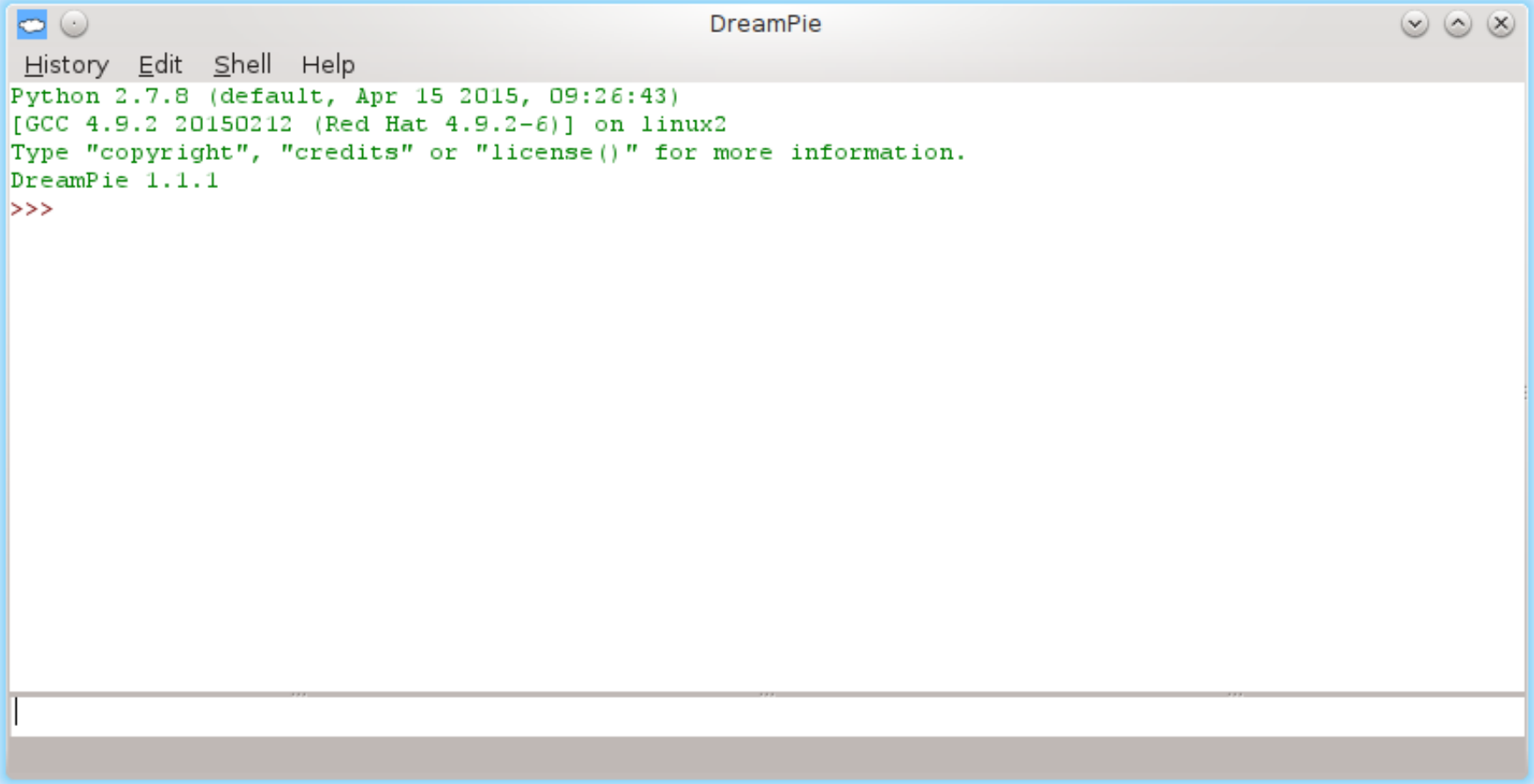


Software design



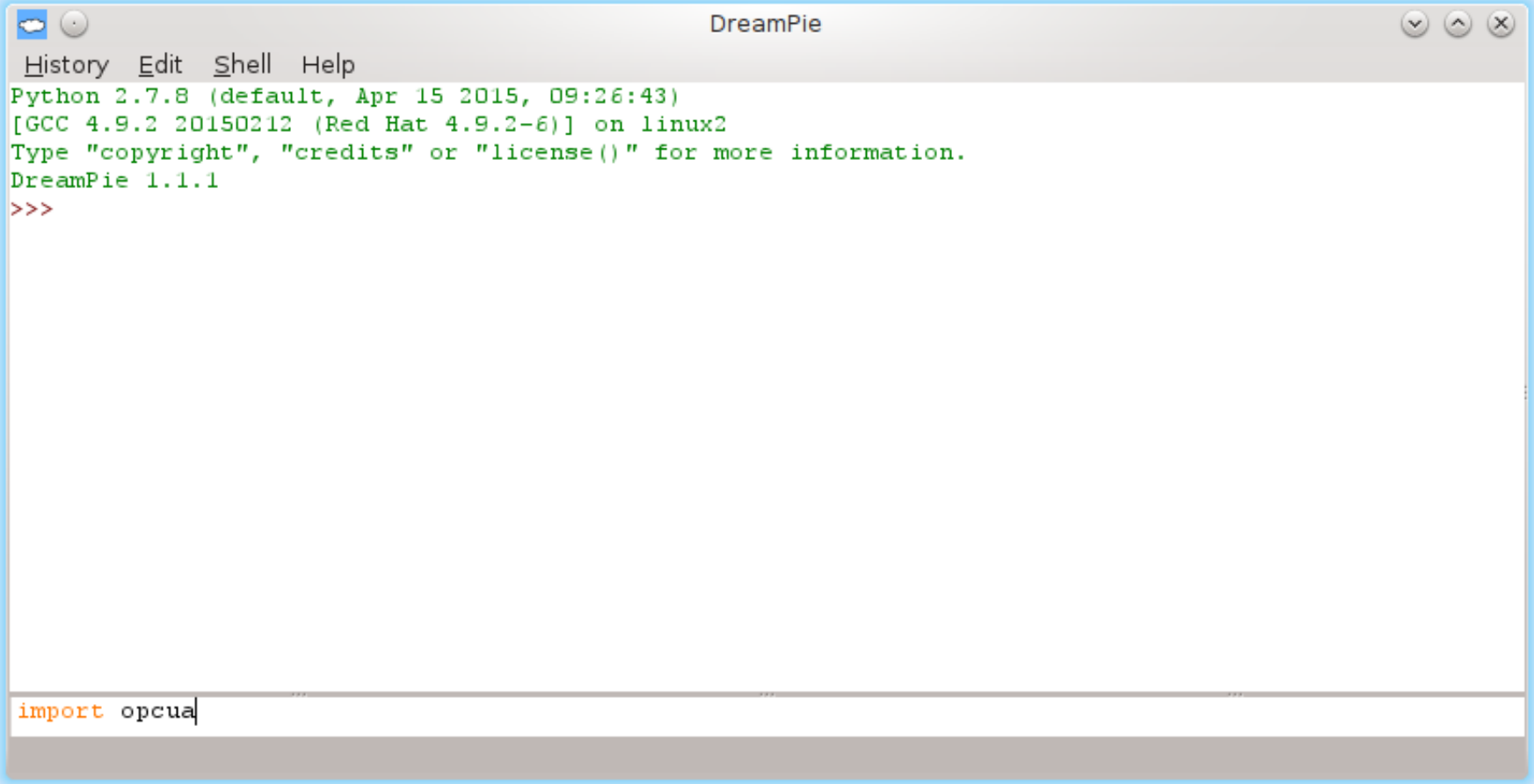
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



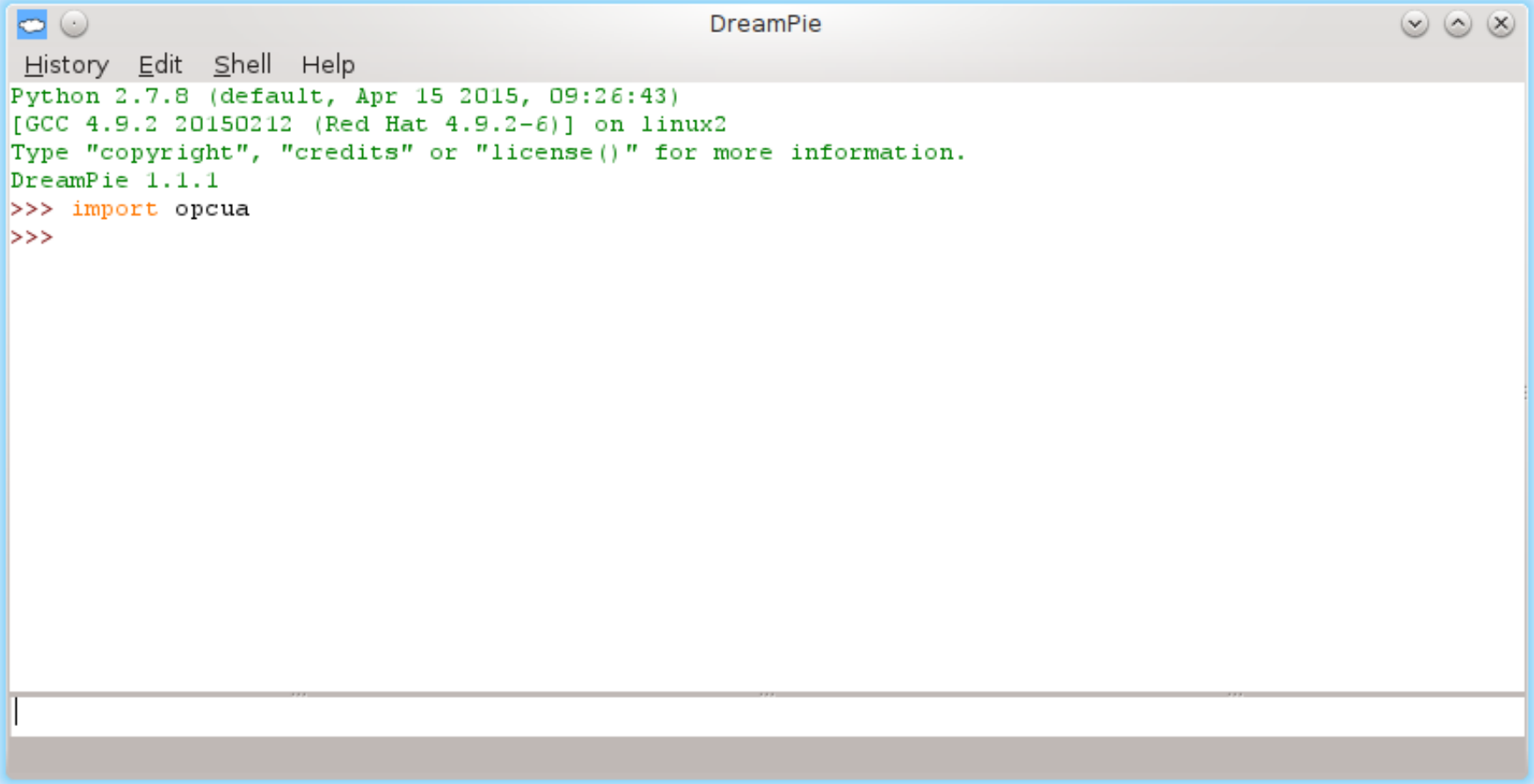
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



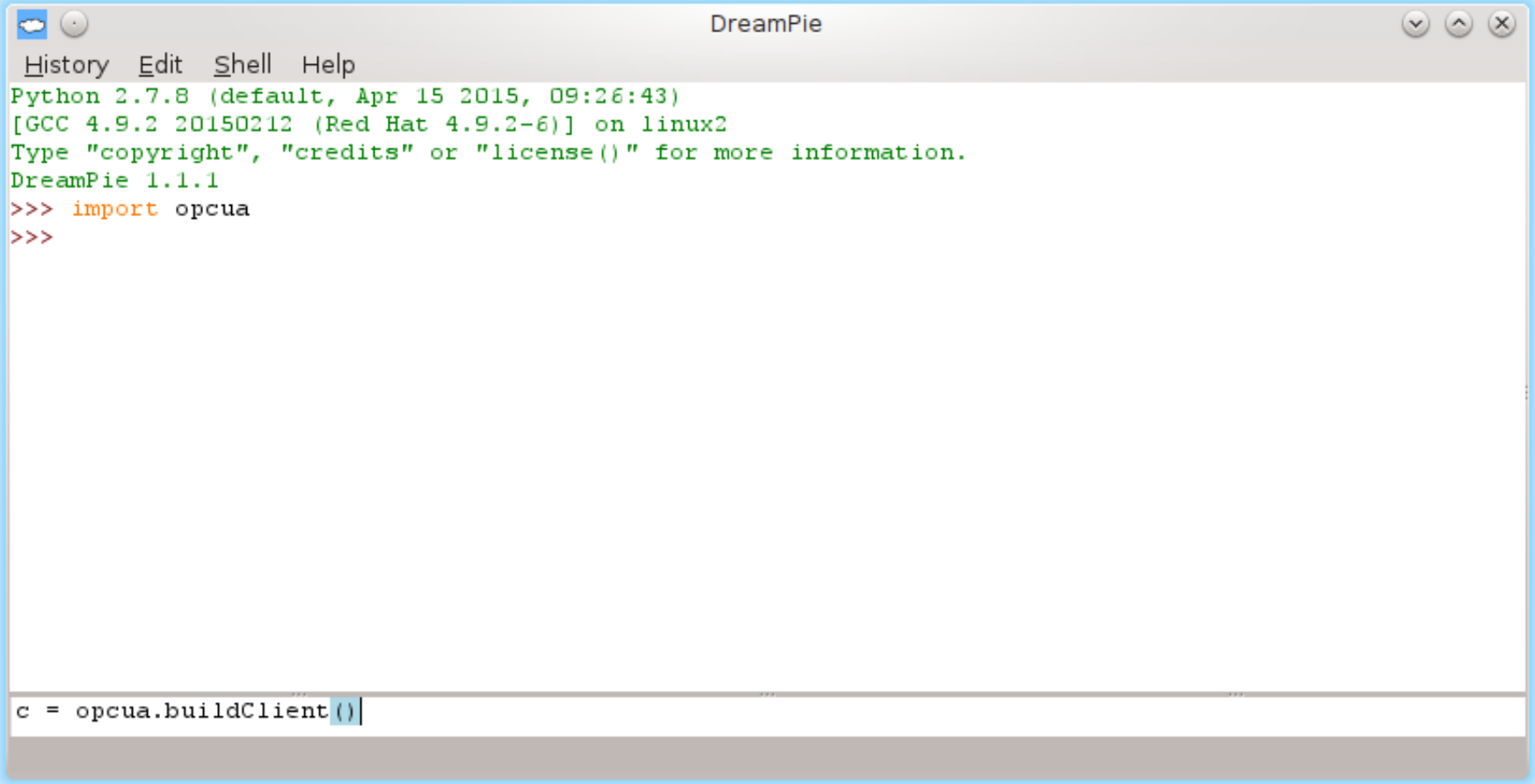
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



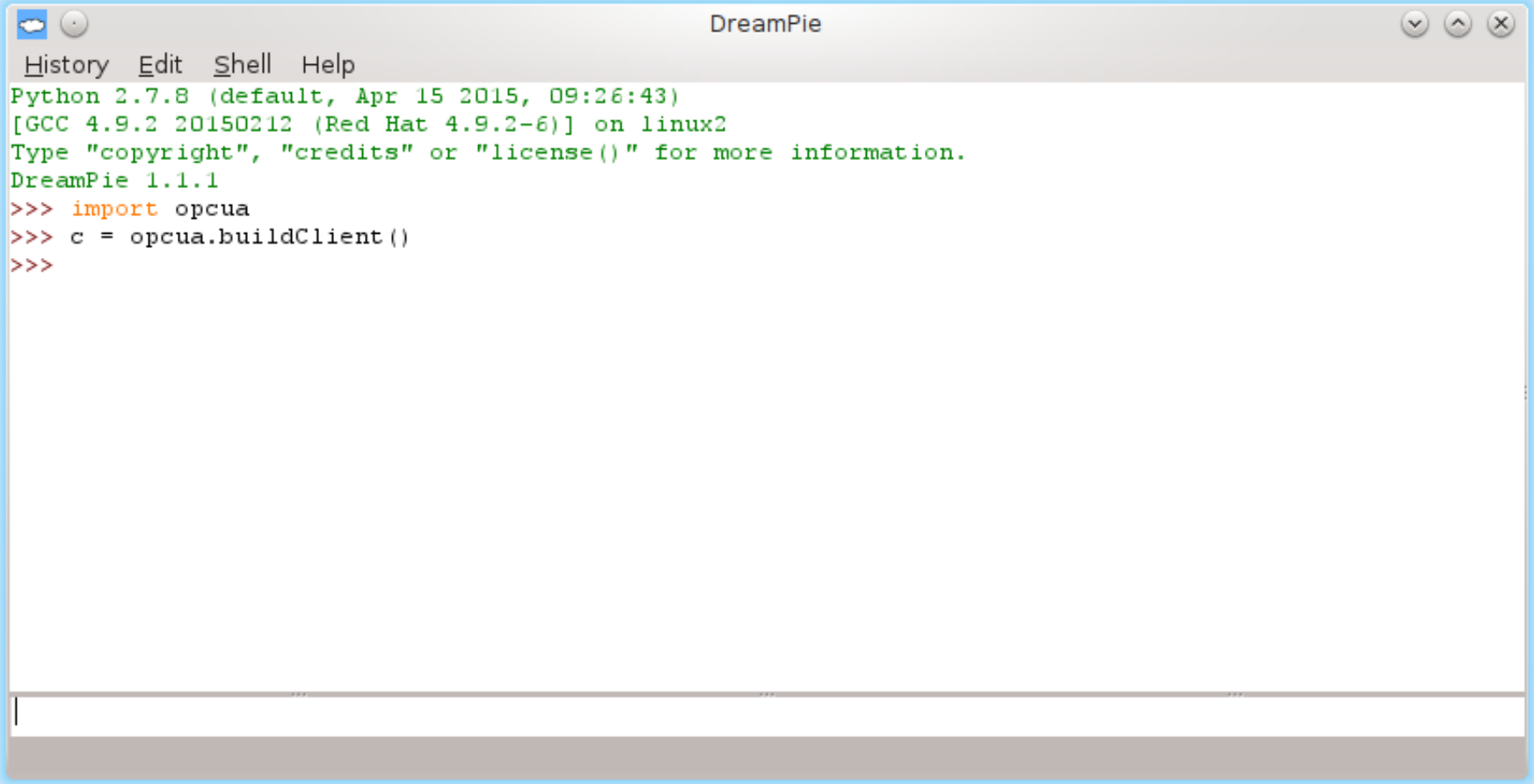
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



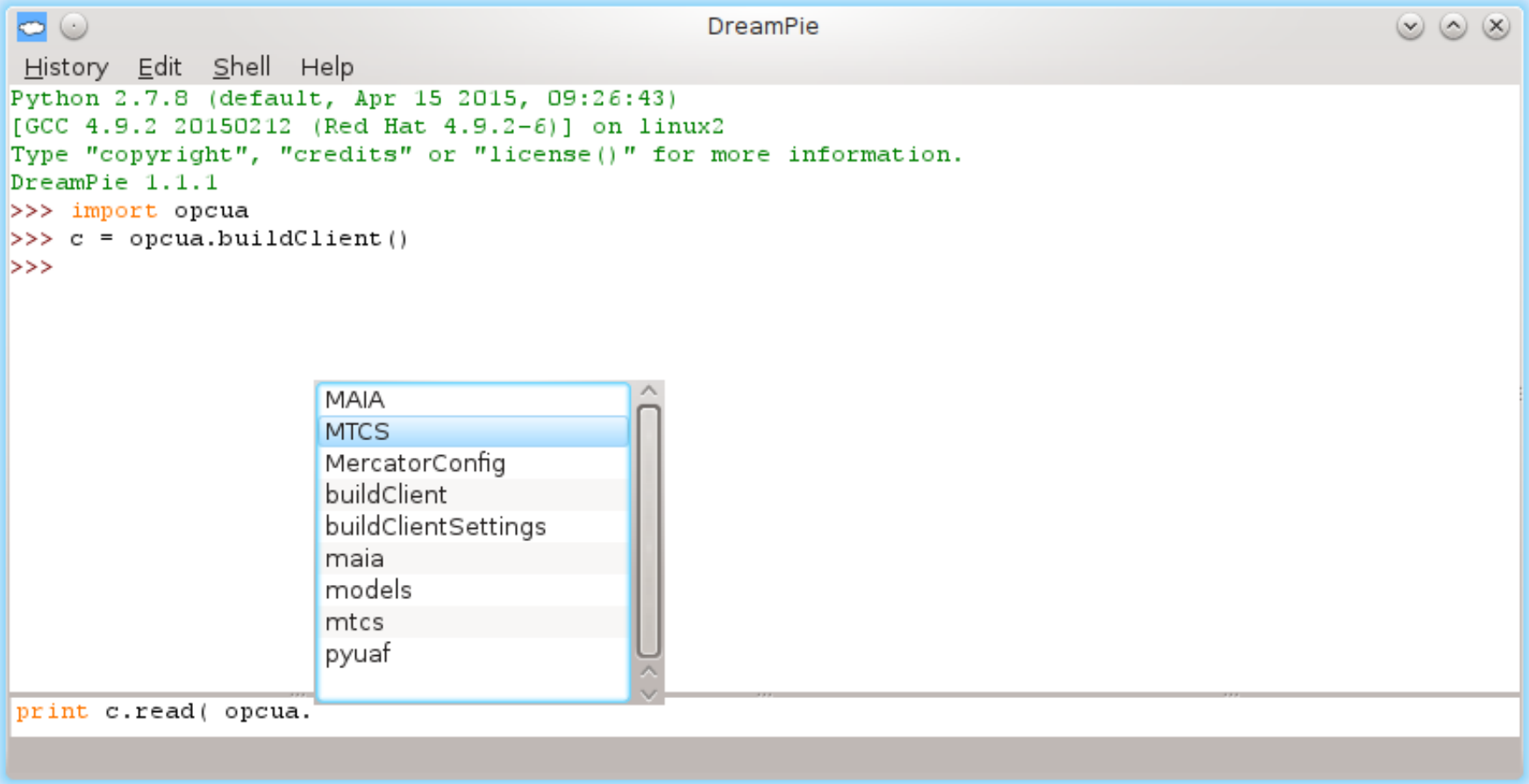
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



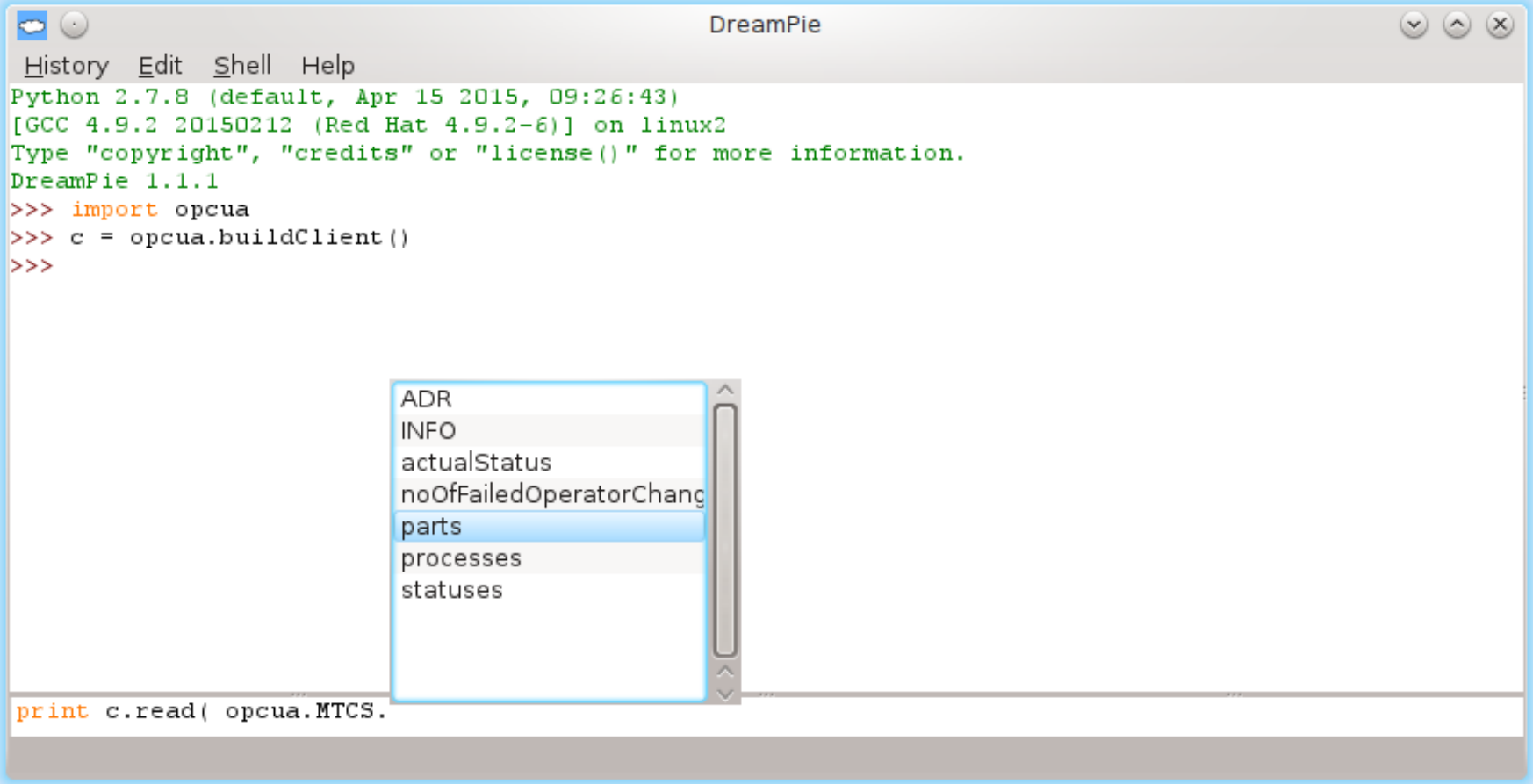
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



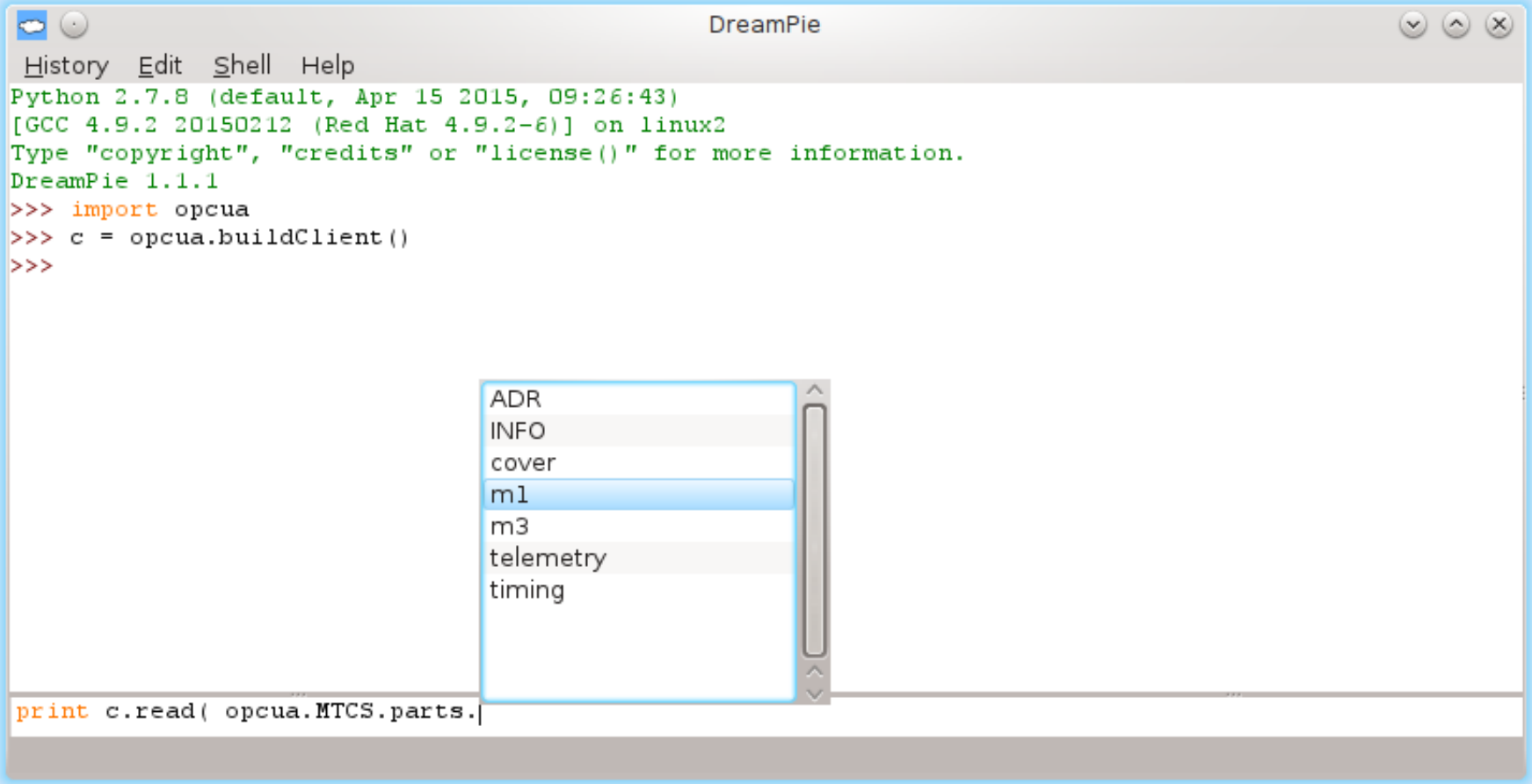
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



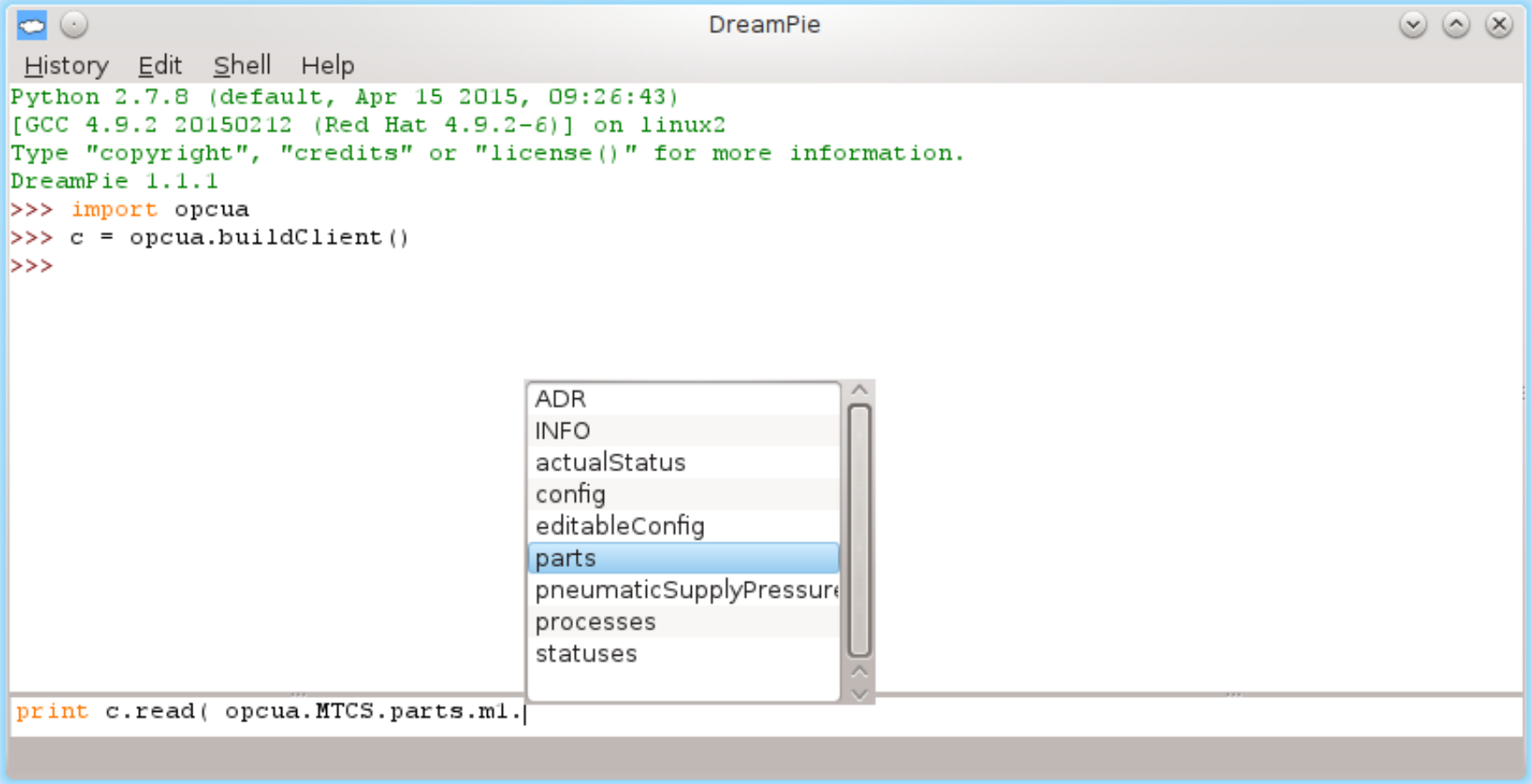
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



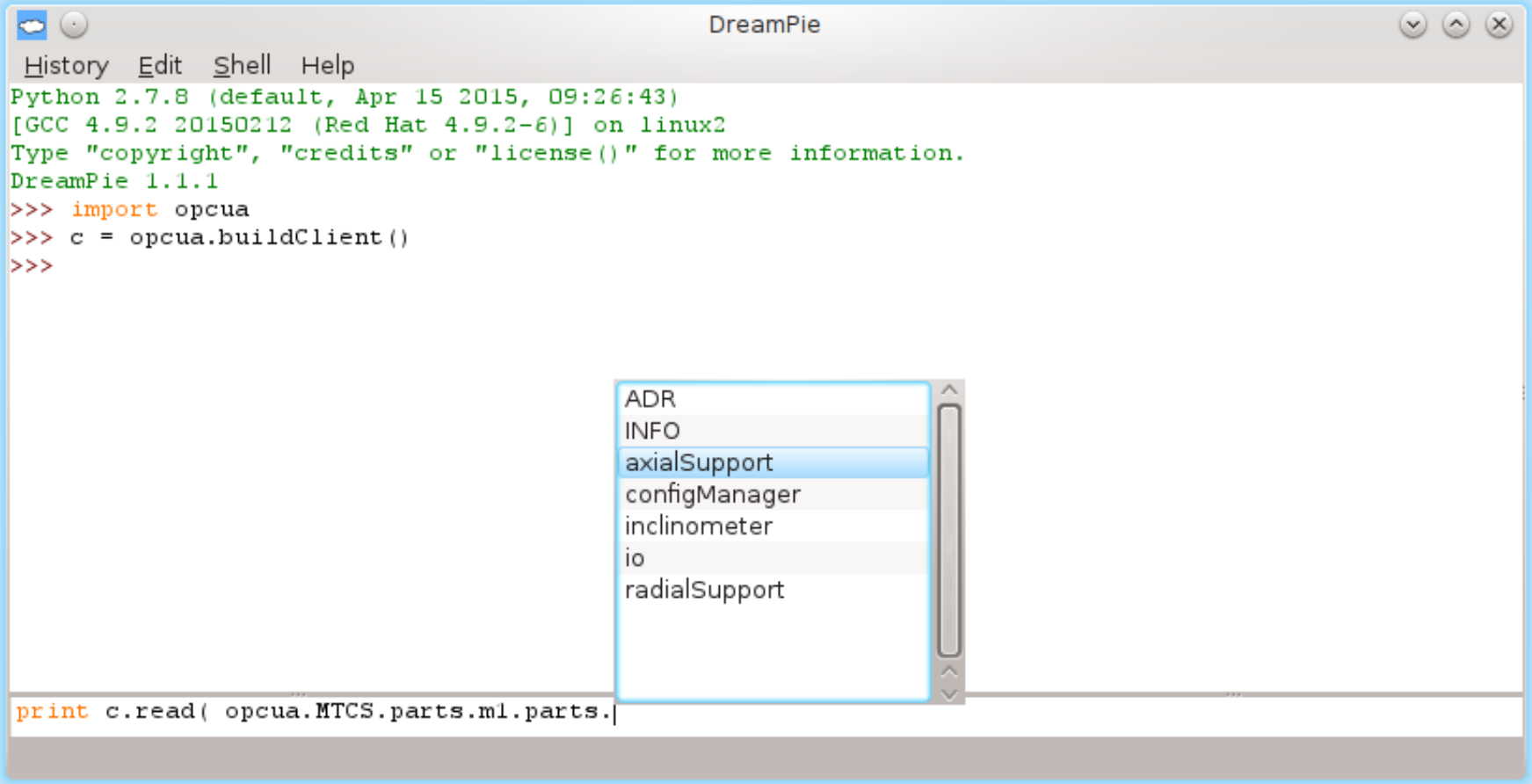
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



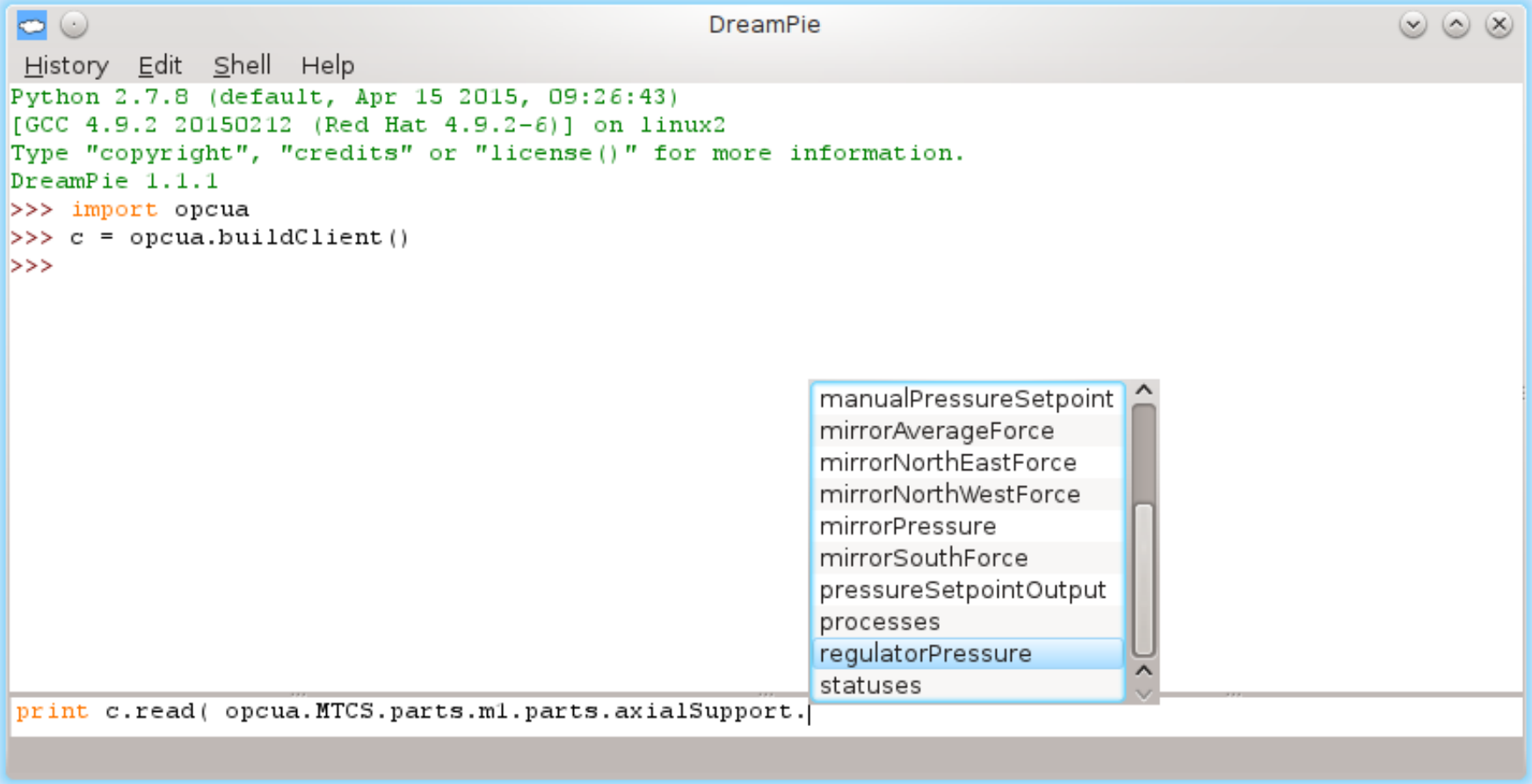
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



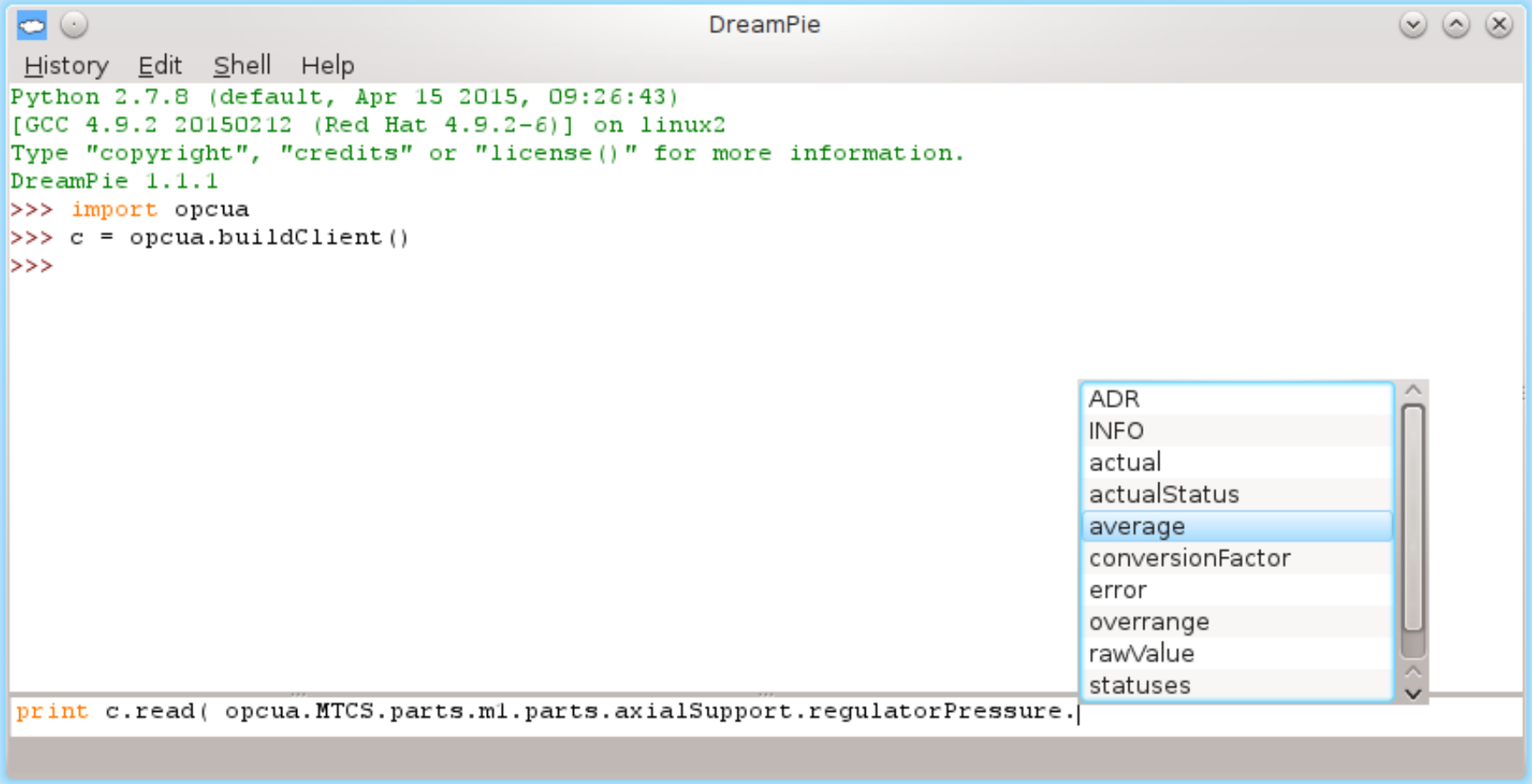
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



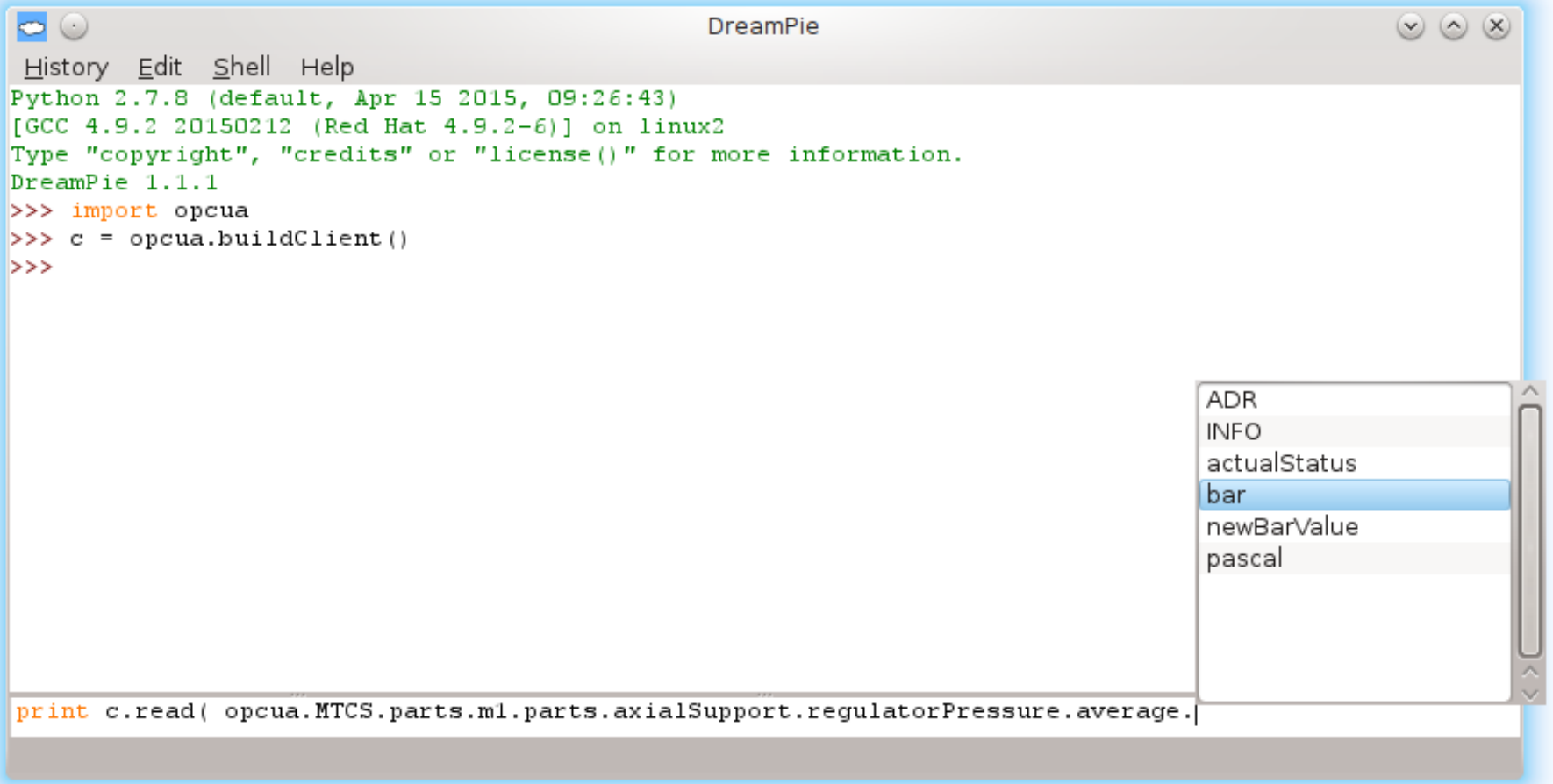
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



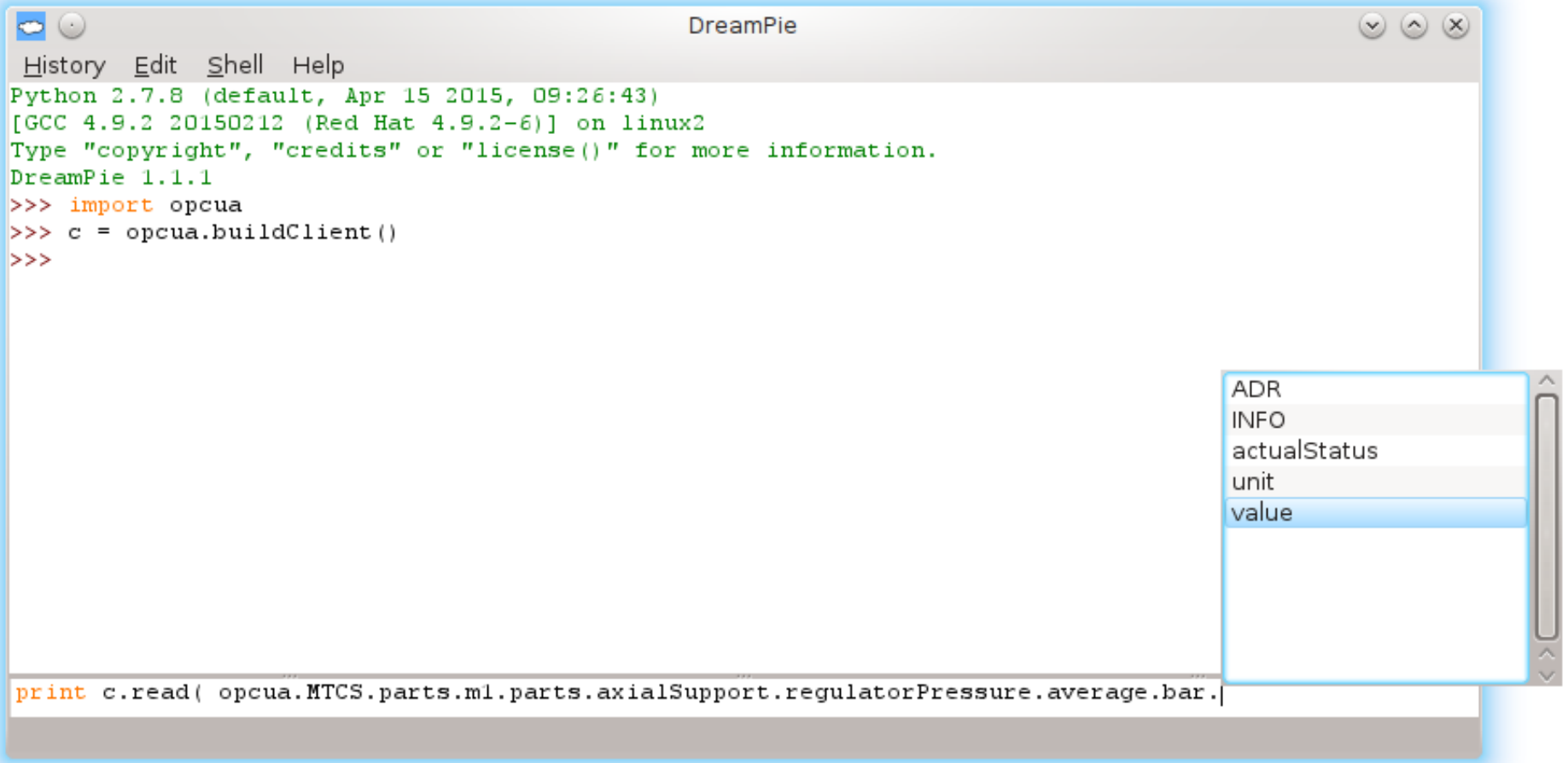
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



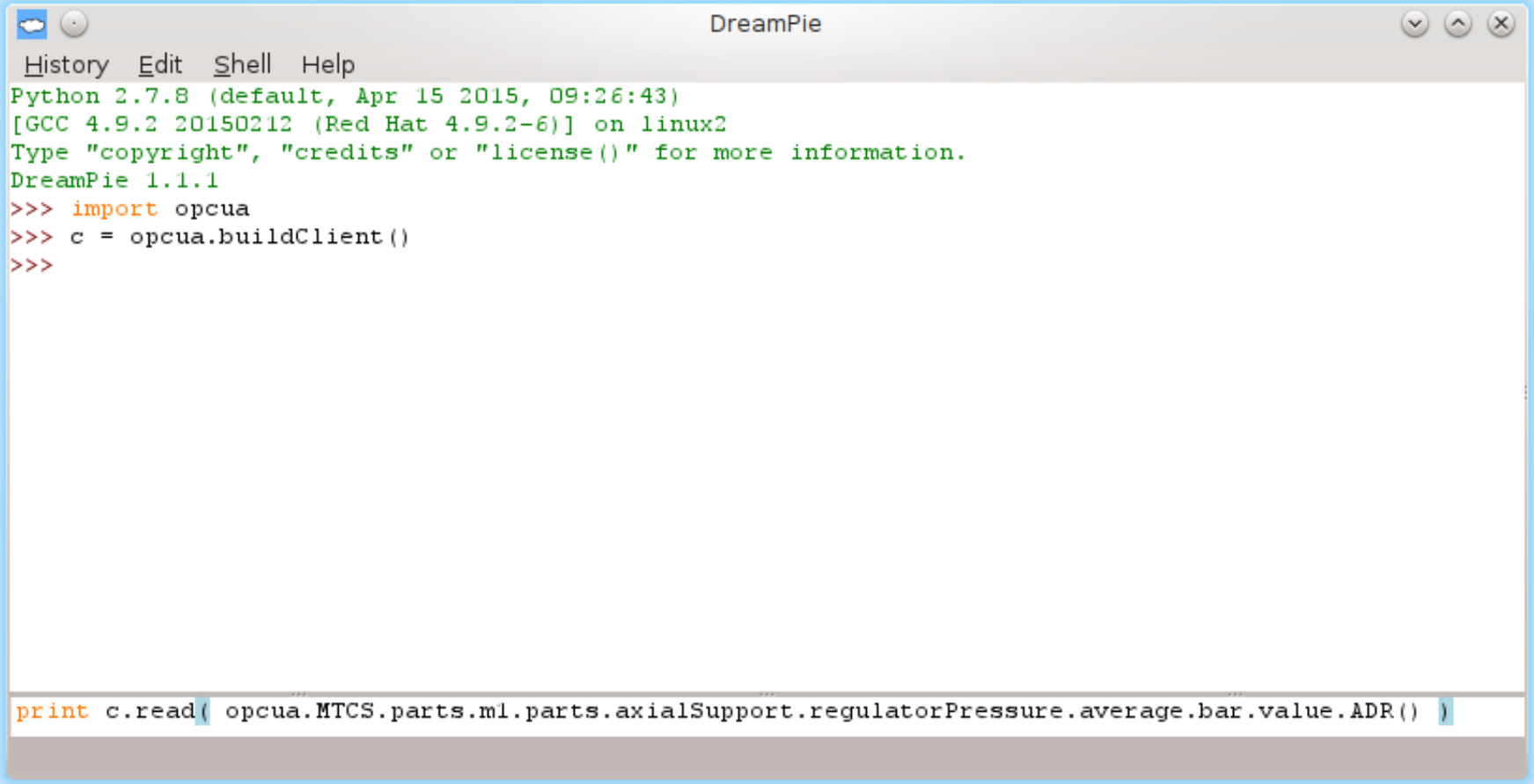
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



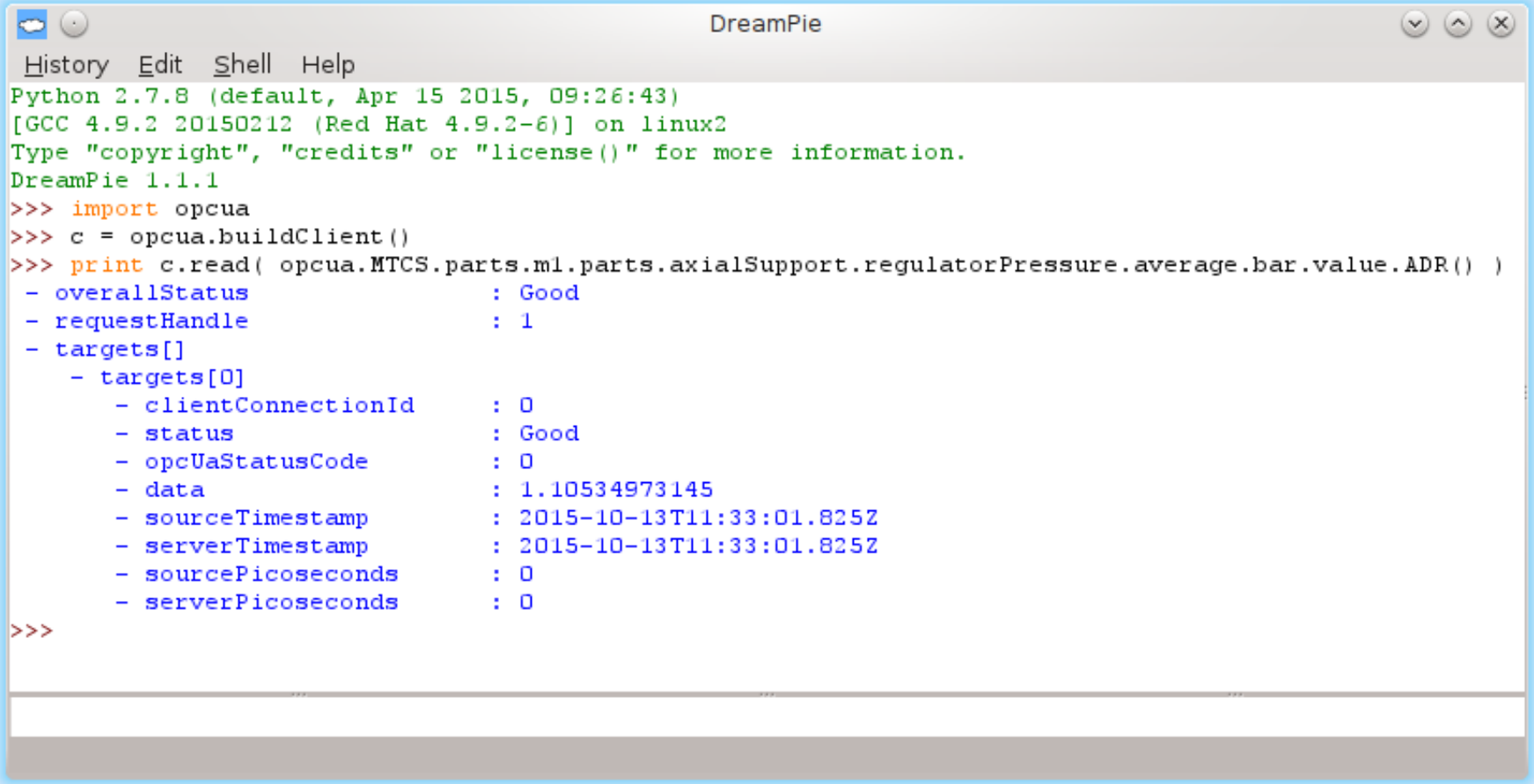
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



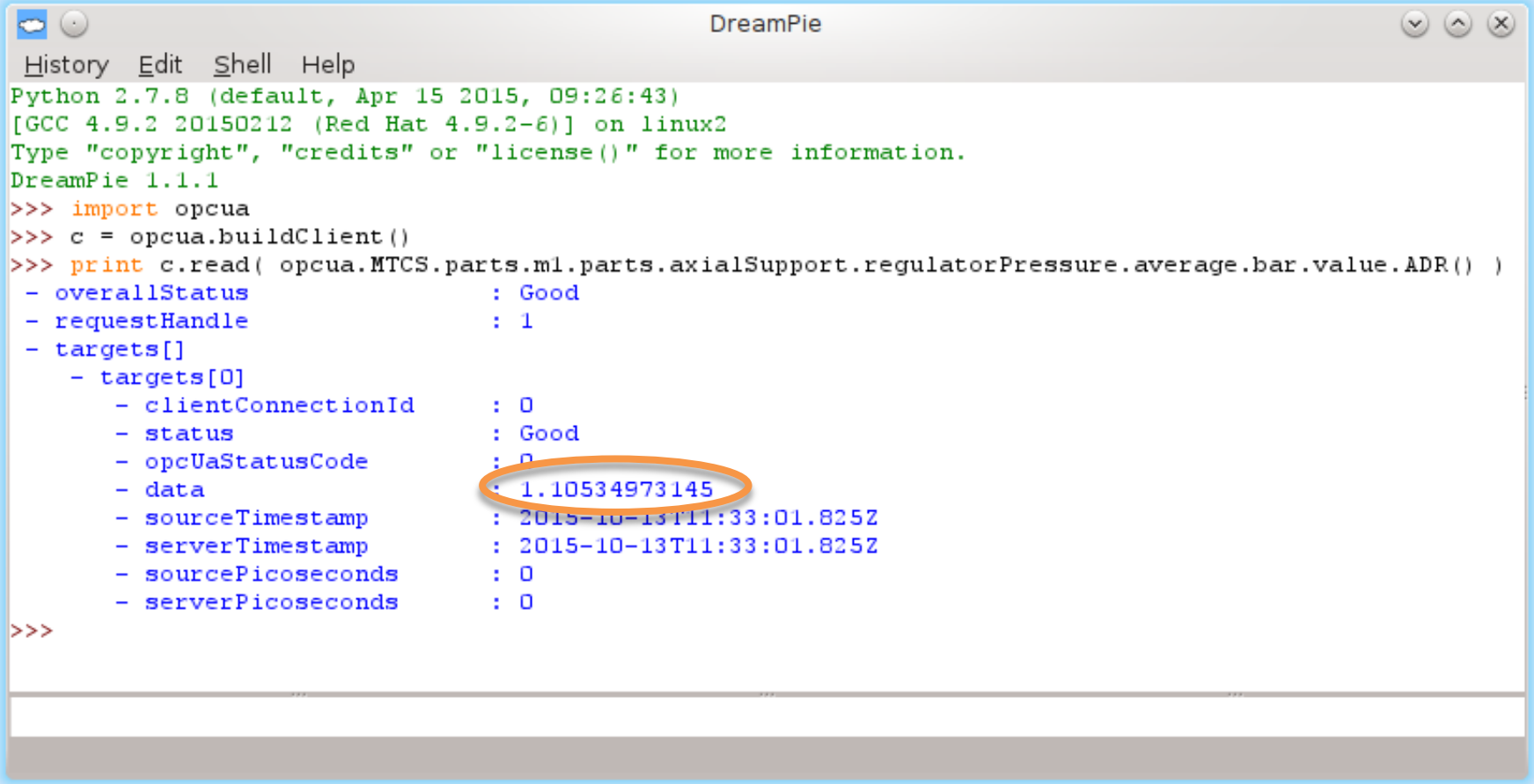
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf



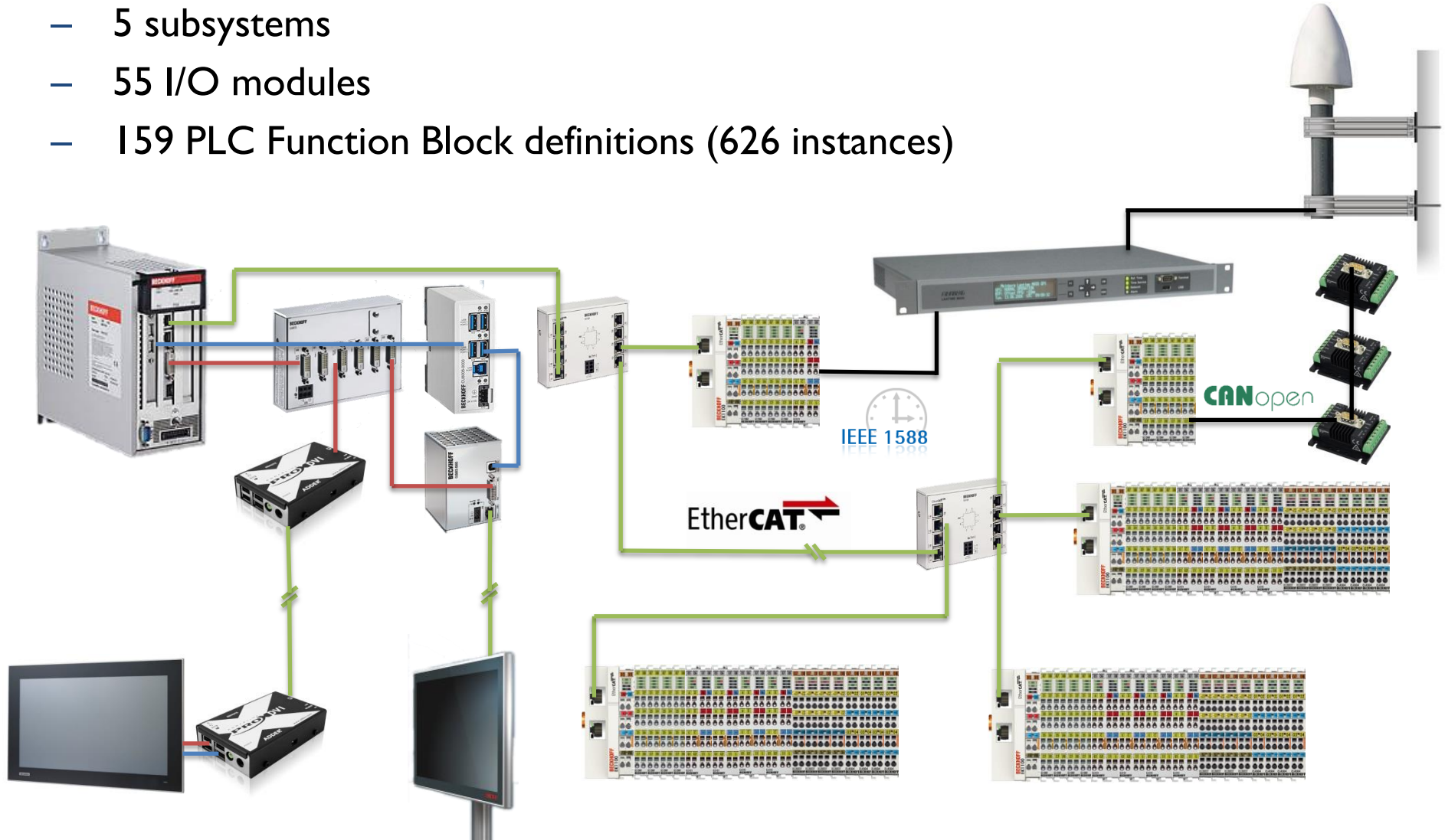
Software design

- Generated Python code (client side)
  - Based on our OPC UA library "UAF": http://github.com/uaf/uaf

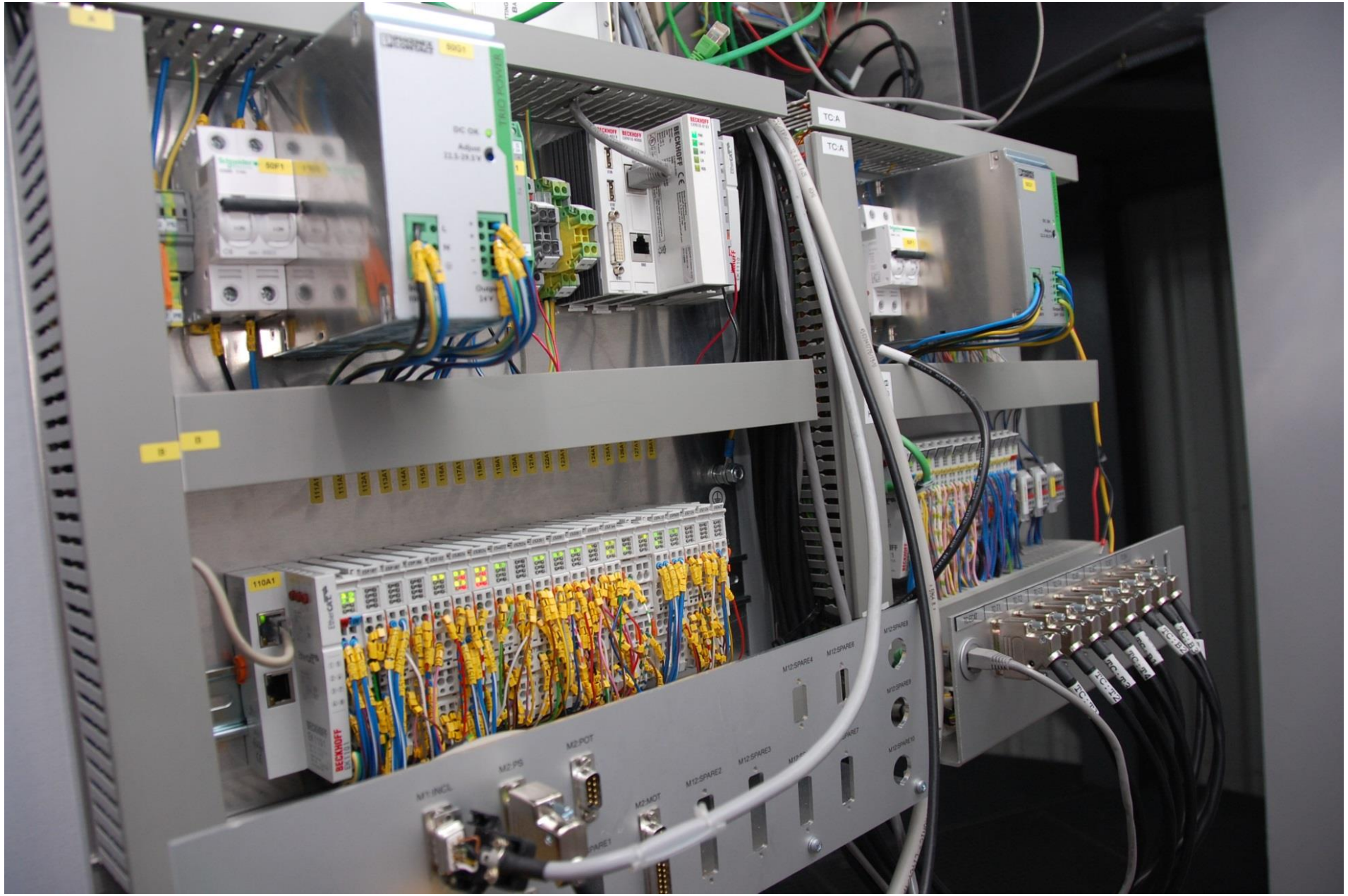


Results

- Currently in operation:
  - 1 PLC
  - 5 subsystems
  - 55 I/O modules
  - 159 PLC Function Block definitions (626 instances)



Results


TC:A
TC:A
110A1
BECKHOFF
M1:INCL
M2:PS
M2:POT
M2:MOT
M12:SPARE2
M12:SPARE3
M12:SPARE4
M12:SPARE6

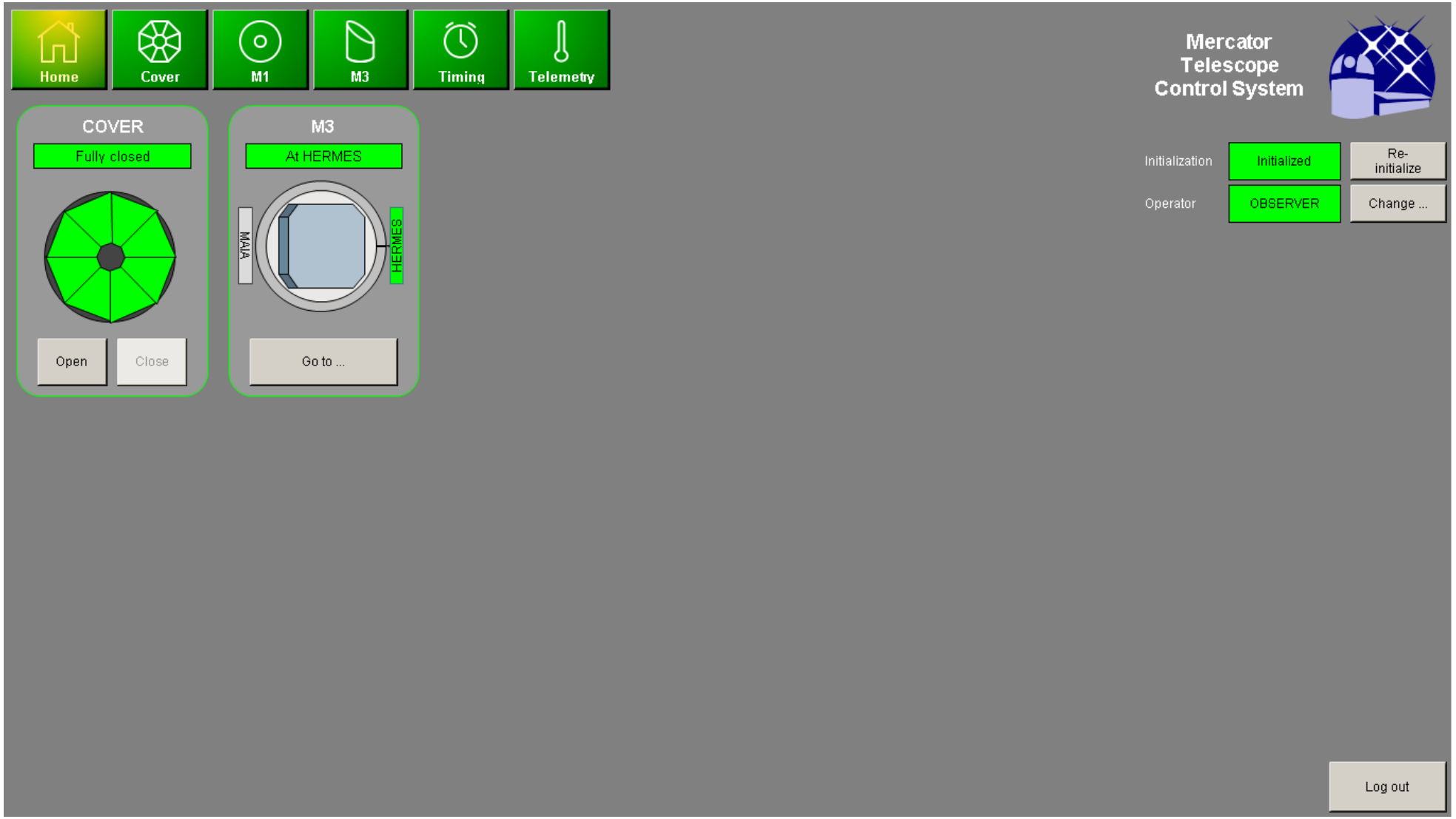
Why semantics matter?

- What are semantic models?
- Where to apply them?
- How to apply them?
- How to build them?
- How to use them?
- Conclusions

Results

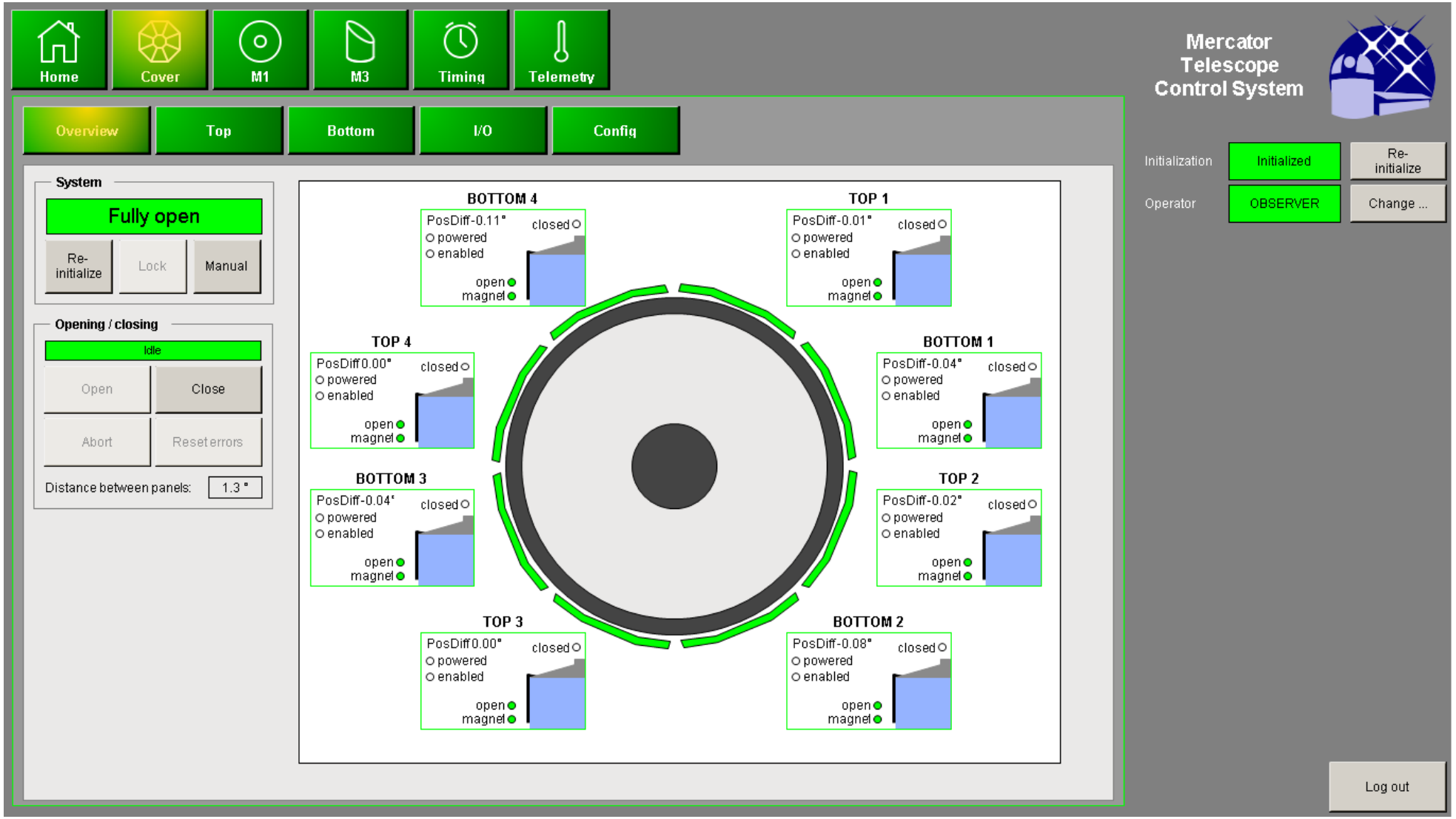
Results

- User Interface (HMI) running **on** the PLC



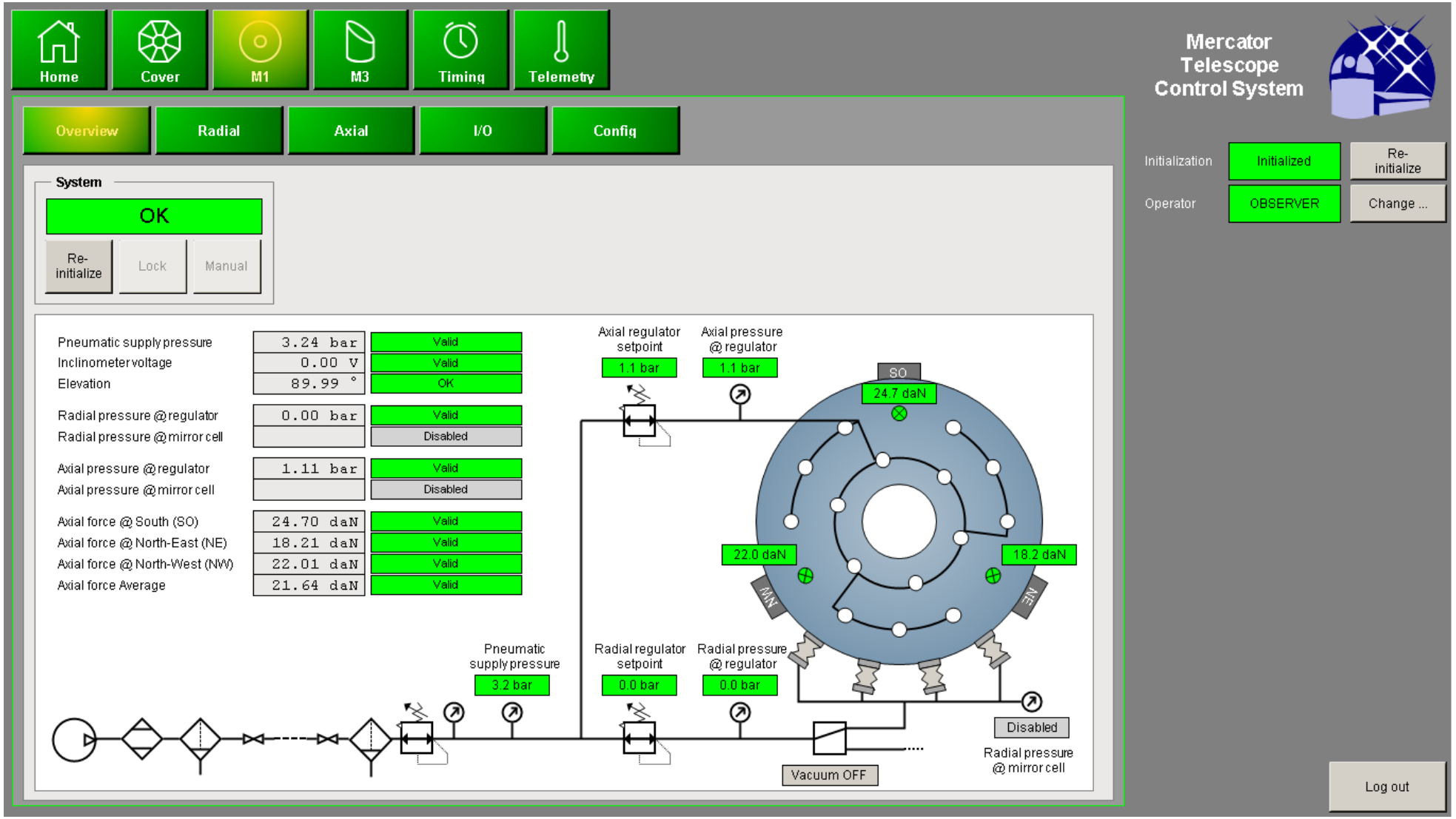
Results

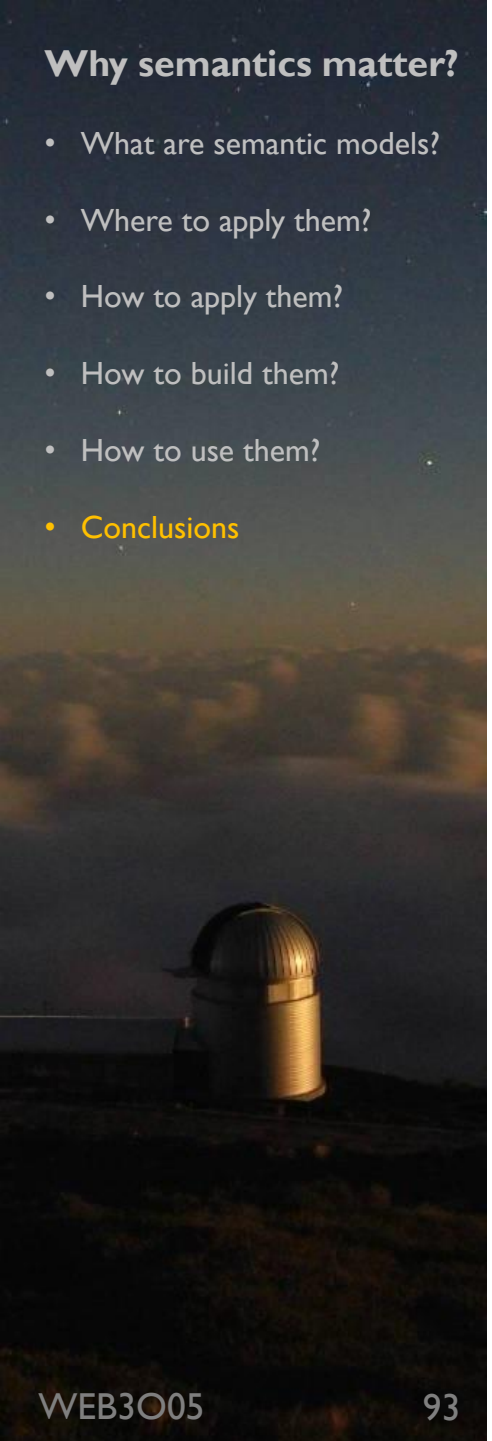
- User Interface (HMI) running **on** the PLC



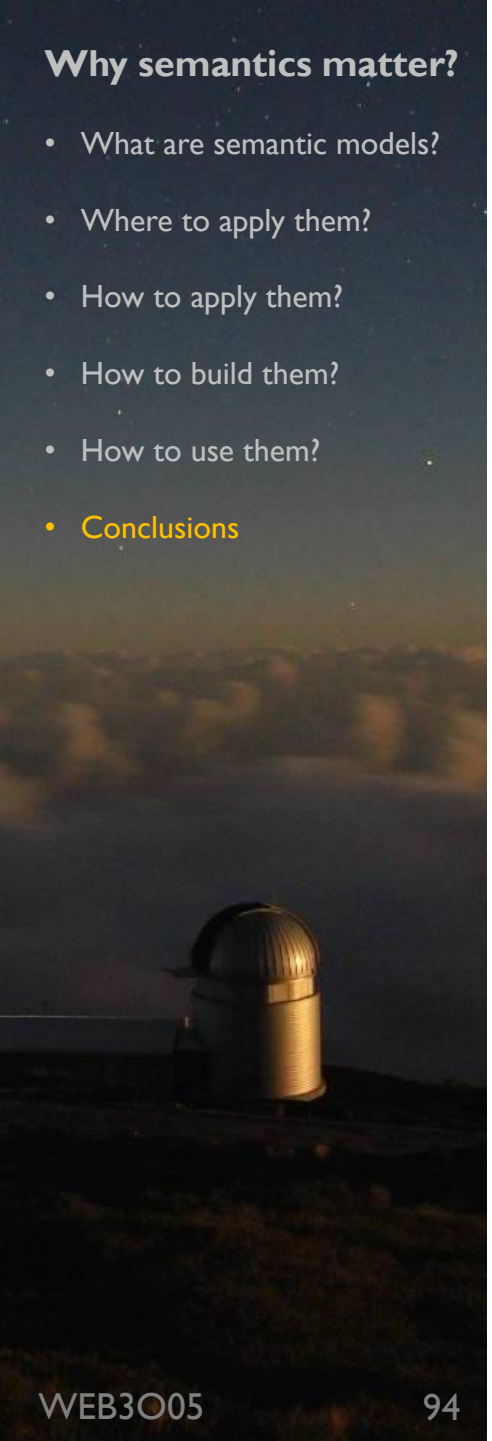
Results

- User Interface (HMI) running **on** the PLC





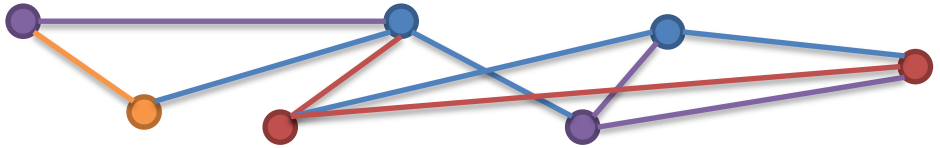
Conclusions



So, why semantics matter?

So, why semantics matter?

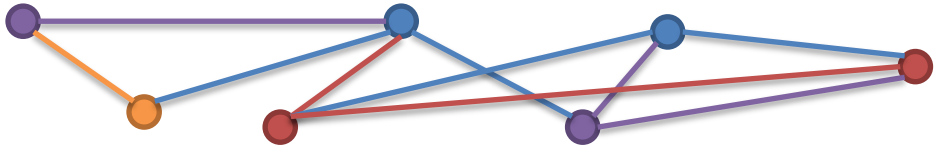
1. Because every piece of information is just one query "away"



➔ **organize, integrate, browse, find (query)** information

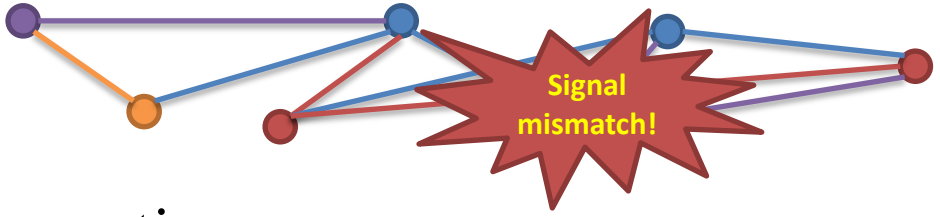
So, why semantics matter?

1. Because every piece of information is just one query "away"



➔ **organize, integrate, browse, find (query)** information

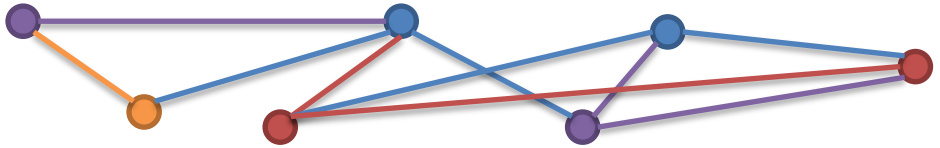
2. Because well defined semantics allow model verification



➔ **verify** information

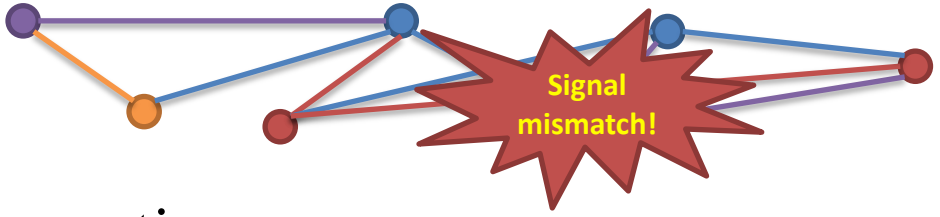
So, why semantics matter?

1. Because every piece of information is just one query "away"



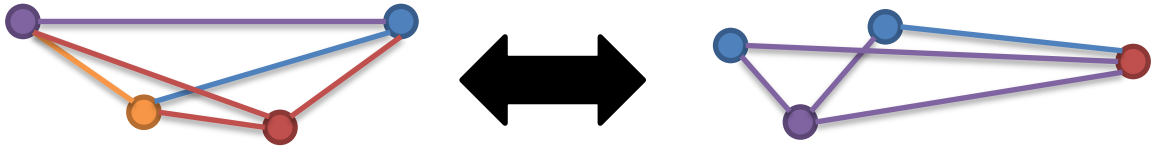
➔ **organize, integrate, browse, find (query)** information

2. Because well defined semantics allow model verification



➔ **verify** information

3. Because they're a key enabling technology for future "smart" systems



➔ **share** information

Thanks!

Any questions?

wim.pessemier@ster.kuleuven.be